UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
TEORÍA DE LA SEÑAL Y COMUNICACIONES

DOCTORAL THESIS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

SPARSE GAUSSIAN PROCESSES FOR LARGE-SCALE MACHINE LEARNING

AUTHOR: MIGUEL LÁZARO GREDILLA
SUPERVISOR: DR. ANÍBAL R. FIGUEIRAS VIDAL

LEGANÉS, MARZO 2010

TESIS DOCTORAL

Sparse Gaussian Processes for Large-Scale Machine Learning

Autor:
Miguel Lázaro Gredilla

Director:
Dr. Aníbal R. Figueiras Vidal

Firma del Tribunal Calificador:

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, a

Agradecimientos

Esta tesis es fruto de mucho tiempo dedicado a ponerme al día en un mundo que me era desconocido, garabatear folios y quedarme absorto mirando una pared, el monitor del laboratorio o la cortina de la ducha. No muy distinto de lo que solía hacer cuando tenía tres años. Supongo que no he cambiado tanto.

En este tiempo he aprendido mucho y aportado un poquito. Y esto ha sido posible gracias a todas las personas que han estado a mi alrededor, han confiado en mí, me han animado y han hecho que esta aventura merezca la pena. Por ello, me gustaría transmitir desde aquí mi agradecimiento a todos ellos:

A Aníbal, que me introdujo en el fascinante mundo de las maquinitas (relativamente) inteligentes y fue el causante de todo esto. Durante estos años, él siempre ha estado ahí, ora supervisando mi trabajo, ora haciéndome saber quién inventó lo mismo hace quince años. Me ha dado ideas, me ha vendido firmas (que afortunadamente aún no ha cobrado). Me ha enseñado turco. Pero lo más importante es la libertad y confianza con la que me ha permitido explorar mis divagaciones, facilitando mi desarrollo no sólo aquí, sino también en otros rincones del mundo. Con esta tesis espero dar respuesta a una antigua pregunta: "¿Investigando...?"

A mis compañeros de laboratorio originales, Luis, Javi y Jaisiel, que hicieron que mis primeros pasos entre estas mismas cuatro paredes fueran divertidos y variados. Hablo de una época en la que los monitores eran CRT y aquí se oía constantemente "eh, caracandao...".

A los tradicionales de estos pasillos, incluidos pasillos paralelos, perpendiculares, y probablemente oblicuos... Gente interesante y agradable, que ha hecho mi vida aquí más cómoda y con la que ha sido muy fácil trabajar. Algunos de ellos: Emilio, siempre dispuesto a escuchar y aportarme su sabiduría. Jero, siempre brillante y con algún comentario en la recámara. Vanessa, siempre resolutiva y útil. Carlos, siempre wey.

Manel, experto en plantas, electromagnetismo y sus interrelaciones. Ángel, con una solución para cada problema. Óscar, que ha aguantado mis insistentes preguntas en los últimos metros de este documento. Darío, melómano, genio y gran persona. Fernando Pérez, habitante intermitente, con el que me he cruzado pocas veces, pero siempre para aprender algo, sea sobre comida japonesa, o emborronando folios.

Desgraciadamente, estas lides son duras y algunos de mis viejos compañeros ya no están con nosotros. De hecho, afortunadamente, han pasado a mejor vida. Me gustaría recordar a Javi, Jaisiel y Manu, a los que se echa mucho de menos, especialmente a la hora de comer. Aquellas conversaciones casi conseguían que pudiera olvidarme de lo que deglutía. ¿Qué fue de la carrera de los números, las vívidas descripciones de Murcia y las discusiones sobre la naturaleza humana y el futuro?

A las siguientes generaciones, compañeros en los laboratorios de tratamiento digital de cosas y la pista de fútbol: Efraín, Luis, Adil, Soufiane...

Aunque es improbable que puedan llegar a leerlo nunca (especialmente en este idioma), es justo agradecer a la gente de Cambridge, por breve que fuera mi estancia allí, su tiempo, dedicación y buenos momentos: Carl, Zoubin y Joaquín, entre los mayores y Finale, Sinead, Jack, Jurgen, Yunus, Marc y Shakir, entre los pequeños. No dejo de sorprenderme de la cantidad de eminencias por metro cuadrado de esa ciudad...

A Nacho y Steven les agradezco su buena disposición y lo fructífero de nuestra relación, que sin duda no terminó cuando dejé Santander. Estupendo lugar, por cierto.

A Nerea, por los buenos momentos y las nuevas experiencias, por hacer mi vida más emocionante e impredecible.

A mis amigos de más de media vida, Expósito, Layla, Álvaro (aka Dennis), Ricardo y Diego. Gracias por no dejar de estar ahí a pesar de la distancia. Y por vuestras visitas, llenas de momentos que no olvidaremos en el resto de nuestras vidas.

A mis abuelos, Porfi y Marcos, por el cariño incondicional recibido a lo largo de toda una vida.

A mis padres, Fernando y Lourdes, que han sido, con diferencia y como de costumbre, las personas con las que más he podido contar y de las que más apoyo he recibido. Me han enseñado a pensar, ser libre y feliz. Me animaron a venir aquí a pesar de resultarles duro y han vivido este tiempo con más intensidad que yo, si cabe.

A mi hermano, que decía que no quería estudiar, y está ya a punto de terminar su segunda ingeniería. Uno de los objetivos de esta tesis (omitido en las subsiguientes secciones) es seguir llevándole ventaja. Aunque sólo pueda ser académicamente. Veremos qué depara el futuro.

A todos, gracias.

Abstract

Gaussian Processes (GPs) are non-parametric, Bayesian models able to achieve state-of-the-art performance in supervised learning tasks such as non-linear regression and classification, thus being used as building blocks for more sophisticated machine learning applications. GPs also enjoy a number of other desirable properties: They are virtually overfitting-free, have sound and convenient model selection procedures, and provide so-called "error bars", i.e., estimations of their predictions' uncertainty.

Unfortunately, full GPs cannot be directly applied to real-world, large-scale data sets due to their high computational cost. For n data samples, training a GP requires $\mathcal{O}(n^3)$ computation time, which renders modern desktop computers unable to handle databases with more than a few thousand instances. Several sparse approximations that scale linearly with the number of data samples have been recently proposed, with the Sparse Pseudo-inputs GP (SPGP) representing the current state of the art. Sparse GP approximations can be used to deal with large databases, but, of course, do not usually achieve the performance of full GPs.

In this thesis we present several novel sparse GP models that compare favorably with SPGP, both in terms of predictive performance and error bar quality. Our models converge to the full GP under some conditions, but our goal is not so much to faithfully approximate full GPs as it is to develop useful models that provide high-quality probabilistic predictions. By doing so, even full GPs are occasionally outperformed.

We provide two broad classes of models: Marginalized Networks (MNs) and Inter-Domain GPs (IDGPs). MNs can be seen as models that lie in between classical Neural Networks (NNs) and full GPs, trying to combine the advantages of both. Though trained differently, when used for prediction they retain the structure of classical NNs, so they can be interpreted as a novel way to train a classical NN, while adding the benefit of input-dependent error bars and overfitting resistance. IDGPs generalize SPGP by allowing the "pseudo-inputs" to lie in a different domain, thus adding extra flexibility and performance. Furthermore, they provide a convenient probabilistic framework in which previous sparse methods can be more easily understood.

All the proposed algorithms are tested and compared with the current state of the art on several standard, large-scale data sets with different properties. Their strengths and weaknesses are also discussed and compared, so that it is easier to select the best suited candidate for each potential application.

Resumen extendido en español

En este resumen se pretende dar cuenta de los objetivos de la presente tesis, así como detallar sus conclusiones y aportaciones originales. Asimismo, se presentan algunas de las futuras líneas de investigación que pueden derivarse de este trabajo.

Introducción y motivación

Los procesos Gaussianos (Gaussian Processes, GPs) son modelos Bayesianos no paramétricos que representan el actual estado del arte en tareas de aprendizaje supervisado tales como regresión y clasificación. Por este motivo, son uno de los bloques básicos usados en la construcción de otros algoritmos de aprendizaje máquina más sofisticados. Asimismo, los GPs tienen una variedad de propiedades muy deseables: Son prácticamente inmunes al sobreajuste, disponen de mecanismos sensatos y cómodos para la selección de modelo y proporcionan las llamadas "barras de error", es decir, son capaces de estimar la incertidumbre de sus propias predicciones.

Desafortunadamente, los GPs completos no pueden aplicarse directamente a bases de datos de gran tamaño, cada vez más frecuentes en la actualidad. Para n muestras, el tiempo de cómputo necesario para entrenar un GP escala como $\mathcal{O}(n^3)$, lo que hace que un ordenador doméstico actual sea incapaz de manejar conjuntos de datos con más de unos pocos miles de muestras. Para solventar este problema se han propuesto recientemente varias aproximaciones "dispersas", que escalan linealmente con el número de muestras. De entre éstas, el método conocido como "procesos Gaussianos dispersos usando pseudo-entradas" (Sparse Pseudo-inputs GP, SPGP), representa el actual estado del arte. Aunque este tipo de aproximaciones dispersas permiten tratar bases de datos mucho mayores, obviamente no alcanzan el rendimiento de los GPs completos.

En esta tesis se introducen varios modelos de GP disperso que presentan un rendimiento mayor que el del SPGP, tanto en cuanto a capacidad predictiva como a calidad de las barras de error. Los modelos propuestos convergen al GP completo que aproximan bajo determinadas condiciones, pero el objetivo de esta tesis no es tanto aproximar fielmente el GP completo original como proporcionar modelos prácticos de alta capacidad predictiva. Tanto es así que, en ocasiones, los nuevos modelos llegan a batir al GP completo que los inspira.

Objetivos y metodología

El objetivo de esta tesis es introducir y poner a prueba a nuevas ideas para la construcción de GPs dispersos que alcancen, hasta cierto punto, las ventajas de los GPs completos y que, al mismo tiempo, sean capaces de manejar conjuntos de datos de gran tamaño.

Nos restringiremos al desarrollo y análisis de modelos que puedan ser entrenados (incluyendo la fase de selección de modelo) en tiempo $\mathcal{O}(m^2 n)$, puedan hacer predicciones probabilísticas en tiempo $\mathcal{O}(m^2)$ por caso de test y requieran un espacio $\mathcal{O}(mn)$, es decir, escalen linealmente con el número de muestras[1]. Existe además un escalado lineal del tiempo de cómputo con la dimensión de los datos de entrada, pero esta dependencia se omite en general. El SPGP será nuestra referencia a batir, ya que tiene precisamente estos órdenes de complejidad y constituye el actual estado del arte en GPs dispersos. Para que las comparaciones sean significativas, nuestros modelos se dimensionarán de manera que el factor multiplicativo del coste computacional coincida con el del SPGP, haciendo así que el tiempo de cómputo necesario para nuestros métodos sea aproximadamente el mismo que el del SPGP.

Las medidas de calidad que usaremos a la hora de evaluar nuestros modelos frente a la referencia serán el error cuadrático (normalizado con respecto a un predictor trivial consistente en la media de los datos de entrenamiento) y la log-probabilidad media de los datos de test. Mientras que la primera medida evalúa sólo la precisión de las predicciones, la segunda evalúa las predicciones probabilísticas de manera global, teniendo

[1] Como comparación, un GP completo se entrena en $\mathcal{O}(n^3)$, puede hacer predicciones probabilísticas en $\mathcal{O}(n^2)$ por caso de test y requiere un espacio $\mathcal{O}(n^2)$.

en cuenta no sólo las medias, sino también las varianzas (es decir las estimaciones de la incertidumbre de las predicciones).

Así pues, todos los algoritmos propuestos serán puestos a prueba y comparados con el SPGP sobre varios conjuntos de datos estándar de diferentes propiedades y de gran tamaño. Se intentarán identificar además las fortalezas y debilidades de cada uno de los métodos, de manera que sea más sencillo elegir el mejor candidato para cada aplicación potencial.

Organización

El Capítulo 1 sirve de introducción a los modelos Bayesianos y en concreto, a los GPs. Contiene además un resumen de los principales hitos en la evolución de los GPs dispersos, haciendo hincapié en las aportaciones de cada uno de estos modelos sobre el anterior. Para clarificar las relaciones entre los diferentes modelos previos, estos se enmarcan, cuando es posible, en el esquema unificador propuesto por Quiñonero-Candela y Rasmussen (2005). Este esquema propone sustituir la vieja interpretación de los GPs disperso como aproximaciones a GPs completos por una nueva en la que se les considera GPs exactos bajo un prior modificado que resulta computacionalmente ventajoso. Todos los modelos de esta tesis se proporcionarán también bajo esta nueva interpretación.

En el Capítulo 2 se introduce el GP de espectro disperso (Sparse Spectrum GP, SSGP), un modelo Bayesiano trigonométrico que puede usarse para aproximar cualquier GP completo estacionario. A diferencia de otros modelos dispersos previos, el SSGP es un proceso auténticamente estacionario (y no solo una aproximación a un proceso estacionario). El SSGP se presenta en primer lugar como una aproximación de Monte Carlo al GP completo, para después proporcionar dos interpretaciones Bayesianas alternativas. Cuando eliminamos el requisito de convergencia al GP completo, la flexibilidad del SSGP se incrementa, proporcionando resultados altamente competitivos en conjuntos de datos de gran tamaño.

En el Capítulo 3 se generaliza el SSGP, insertándolo en una clase de modelos Bayesianos más amplia, con la estructura de un modelo lineal generalizado en el que los "pesos de salida" han sido marginalizados. Nos referiremos a esta clase como redes

marginalizadas (Marginalized Networks, MNs). Se investigan los problemas resultantes de la aplicación directa de las MNs a problemas de regresión y se proponen dos maneras diferentes de evitarlos: Ruido acotado y mezcla de redes. Se comprueban las ventajas de estas MNs mejoradas en problemas de gran tamaño y se compara con el anterior SSGP (así como con el SPGP). Finalmente, se muestra como se puede aprovechar la estructura de estas redes para reducir linealmente la dimensión de los datos de entrada (manteniendo su capacidad de predictiva sobre la variable de salida).

En el Capítulo 4 se extienden los GPs a múltiples dominios (Inter-domain GPs, IDGPs). Definiendo un GP sobre más de un dominio, es posible extender el SPGP y situar las pseudo-entradas[2] en otros dominios. Esto tiene dos efectos de interés: Por una parte, desacopla la forma de las funciones de base de la forma de la función de covarianza, y por otra, añade mayor flexibilidad y capacidad expresiva al modelo, mejorando su rendimiento. Se trata de un marco general que incluye a otros modelos desarrollados previamente, tales como los GPs multi-escala de Walder et al. (2008), aportando una nueva perspectiva sobre su significado e interpretación. Los IDGPs también pueden usarse para otros propósitos no relacionados con la inferencia computacionalmente eficiente, tales como inferencia entre dominios o imponer restricciones sobre la función latente, pero no exploraremos esas posibilidades.

En el Capítulo 5 se investigan varias extensiones de las ideas previas. En primer lugar, se expresan los perceptrones multicapa como MNs y se comprueba su rendimiento. A continuación, se proporcionan los detalles necesarios para extender los GPs dispersos a verosimilitudes no Gaussianas sin empeorar su eficiencia computacional. Este proceso se describe de forma general para cualquier modelo con una determinada estructura en su matriz de covarianza, lo que permite aplicar todos los modelos presentados en esta tesis a clasificación y regresión robusta. Desarrollaremos ambos casos y los aplicaremos en los conjuntos de datos correspondientes.

El Capítulo 6 concluye esta tesis con un resumen de las contribuciones realizadas, una comparativa de los modelos propuestos y un breve resumen de posibles líneas futuras.

Finalmente, se incluyen seis apéndices que sirven de referencia a lo largo de esta tesis. En ellos se incluyen relaciones básicas de álgebra matricial, identidades útiles

[2] Al situarlas en otros dominios distintos al de entrada, nos referiremos a ellas como pseudo-características.

para el manejo de distribuciones Gaussianas, demostraciones de convergencia de algunos de los modelos en el límite infinito y los detalles necesarios para una implementación eficiente de los algoritmos propuestos.

Conclusiones y líneas futuras

En esta tesis se han desarrollado y evaluado varios modelos de GP disperso junto con diferentes estrategias de selección de modelo. Estos GPs dispersos intentan conservar las ventajas de los GP completos (gran precisión, predicciones probabilísticas, inmunidad al sobreajuste), al tiempo que reducen sensiblemente los requisitos cómputo y memoria de $\mathcal{O}(n^3)$ y $\mathcal{O}(n^2)$ a $\mathcal{O}(m^2n)$ y $\mathcal{O}(mn)$, respectivamente. Las comparaciones con SPGP, el actual estado del arte, muestran que los métodos propuestos proporcionan una mejora significativa en la práctica y pueden por tanto resultar útiles para atacar bases de datos a gran escala. Aunque este trabajo ha estado principalmente centrado en modelos de regresión con ruido Gaussiano, hemos mostrado en el Capítulo 5 como estos modelos pueden extenderse directamente para llevar a cabo regresión robusta, clasificación, etc.

A continuación se resumen las principales contribuciones de esta tesis, se contrastan las ventajas y desventajas de los diferentes algoritmos introducidos en un cuadro comparativo y se mencionan posibles futuras líneas de trabajo.

Aportaciones originales

- **GP de espectro disperso (SSGP).** En el Capítulo 2 se ha introducido el SSGP, en el que se explota la interpretación espectral de los GPs para conseguir una matriz de covarianza dispersa (lo que se traduce directamente en mayor rendimiento computacional). El SSGP tiene varias propiedades que lo distinguen de otros modelos dispersos: Tiene una covarianza auténticamente estacionaria (a diferencia de la mayor parte de los GP dispersos, que sólo aproximan dicha estacionariedad), no tiene parámetros espaciales en el dominio de entrada (como podrían ser las pseudo-entradas del SPGP, o el conjunto activo de otros modelos), y usa funciones base globales, periódicas (en contraste con las bases locales

usadas por la mayor parte de los GP locales). Se han proporcionado tres interpretaciones equivalentes del SSGP, una de las cuales muestra su correspondencia con un modelo lineal generalizado de bases tipo coseno en el que las fases han sido marginalizadas.

Se han propuesto dos estrategias alternativas para seleccionar las llamadas "muestras espectrales" (es decir, las frecuencias de las bases usadas en la aproximación):

(a) **Muestras espectrales fijas:** Si se generan aleatoriamente a partir de una densidad de probabilidad $p(\mathbf{s}_r)$ y se mantienen fijas, el SSGP aproxima un GP completo cuya función de covarianza es proporcional a la transformada de Fourier inversa de $p(\mathbf{s}_r)$. La convergencia al GP completo se alcanza cuando el número de muestras espectrales tiende a infinito. Tan sólo se requiere seleccionar un pequeño número de hiperparámetros (los mismos que para un GP completo), por lo que la selección de modelo es rápida y se evita el sobreajuste. La contrapartida a estas ventajas es un rendimiento más limitado.

(b) **Muestras espectrales seleccionables:** Si aprendemos las muestras espectrales (además de los hiperparámetros), la precisión de las predicciones aumenta sensiblemente. Al introducirse este grado de libertad adicional, cuando el número de muestras espectrales es grande, existe un riesgo de sobreajuste. Hemos mostrado empíricamente que este sobreajuste (en el sentido explicado en la Sección 2.5) es rara vez un problema (probablemente debido a la propiedad de marginalización de fase antes mencionada), pero puede dar lugar ocasionalmente a varianzas predictivas pobres (es decir, el modelo puede mostrar un exceso de confianza en sus propias predicciones).

- **Redes marginalizadas (MNs).** En el Capítulo 3 se introducen las MNs junto a dos estrategias de reducción de sobreajuste. Las MNs son, esencialmente, modelos lineales generalizados en los que los pesos de salida han sido marginalizados. Los pesos de entrada pueden generarse a partir de una determinada distribución de probabilidad y fijarse, o aprenderse junto a los hiperparámetros de potencia de ruido y de señal. Al igual que con el SSGP, fijar los pesos de entrada evita completamente el sobreajuste, pero se requieren más funciones de base para obtener un alto rendimiento (es decir, el modelo es menos disperso). Cuando se aprenden

los pesos de entrada, sólo se necesita un número moderado de funciones de base, pero pueden aparecer problemas de sobreajuste. Para evitarlos, se proponen dos nuevas estrategias: Ruido acotado y mezcla de redes.

(a) El **ruido acotado** es una estrategia sencilla y fácil de aplicar: Se estima la potencia del ruido presente en el conjunto de datos y después se utiliza este valor para acotar inferiormente el hiperparámetro correspondiente. Hemos argumentado teóricamente y comprobado experimentalmente que esto ayuda a reducir el sobreajuste. Aunque este método consigue su propósito, dando lugar a predicciones de buena calidad (medias predictivas precisas), puede producir varianzas predictivas pobres.

(b) La **mezcla de redes** es un procedimiento análogo al "bagging" (ver por ejemplo Breiman (1996)), pero en este caso la diversidad se introduce utilizando diferentes inicializaciones aleatorias, en lugar de utilizar diferentes subconjuntos de los datos. Hemos mostrado empíricamente que esta estrategia mejora tanto las medias como las varianzas predictivas, produciendo en conjunto unas predicciones probabilísticas de alta calidad.

Es posible expresar el SSGP como un tipo particular de MN con restricciones adicionales sobre los pesos de entrada. Estas restricciones (que dan lugar al efecto de marginalización de fase) parecen ser la causa de la resistencia al sobreajuste inherente al SSGP. Como se muestra en el Capítulo 2, el SSGP puede proporcionar buenos resultados incluso sin recurrir al ruido acotado o la mezcla de redes. A pesar de esto, si el número de muestras espectrales no es pequeño en comparación con el número de muestras de entrada, se recomienda aplicar alguna de estas técnicas para reducir el riesgo de sobreajuste.

También se ha mostrado explícitamente como la estructura de las MNs permite una reducción lineal de la dimensión de los datos de entrada (manteniendo su capacidad predictiva) a un coste muy reducido. Esto se puede conseguir:

(a) **Forzando una reducción de dimensión en el propio diseño**, es decir, aprendiendo una proyección lineal de los datos de entrada en un espacio de una dimensión menor, preespecificada.

(b) **Descubriendo la dimensión intrínseca de la función que se está aprendiendo** a través de la descomposición en valores singulares de la matriz

de pesos de entrada tras el entrenamiento, lo cual requiere un tiempo de cómputo de sólo $\mathcal{O}(mD^2)$.

- **GPs inter-dominio (IDGPs).** En el Capítulo 4 se han extendido los GPs a varios dominios linealmente relacionados, mostrando como era posible relacionar variables que se encontraban en diferentes dominios para hacer inferencia conjunta. Se explotó esta posibilidad para extender el SPGP, permitiendo a las pseudo-entradas situarse en un dominio diferente al de los los datos de entrada. Al no encontrarse en el dominio de entrada nos referimos a ellas como "pseudo-características", enfatizando así como cada una de ellas nos informa sobre una característica del conjunto de datos sobre el que se trabaja. Algunas aproximaciones existentes como el propio SPGP o los GPs multi-escala de Walder et al. (2008) pueden interpretarse como particularizaciones de este modelo. Expresar una determinada aproximación como un tipo de IDGP puede simplificarnos su interpretación y darnos más detalles sobre él. Por ejemplo, en el caso de los GPs multi-escala, se obtiene de manera natural la mejora de la varianza a priori que se introdujo de manera post-hoc en el modelo original. Y lo que es más relevante, se muestra que las condiciones propuestas en los GPs multi-escala no son suficientes para garantizar una interpretación probabilística del modelo. Las condiciones necesarias para que esta interpretación sea posible se obtienen directamente de considerarlos como IDGPs.

  Los IDGPs se pueden aplicar a cualquier dominio que consista en una transformación lineal del espacio de entrada (incluyendo convoluciones, integrales y derivadas). En este trabajo hemos desarrollado

  (a) IDGPs para (una versión "borrosa" de) el dominio de la frecuencia (**Frequency Inducing-Features GP, FIFGP**)

  (b) IDGPs para un dominio combinado de tiempo y frecuencia (**Time-Frequency Inducing-Features GP, TFIFGP**).

  Ambos métodos comparten las principales propiedades del SPGP, pero resultan en un rendimiento mayor (en términos de capacidad predictiva) en los conjuntos de datos considerados.

- Extensión de las MNs al caso de los perceptrones multicapa y de todos los modelos vistos al caso de verosimilitudes no Gaussianas. En el Capítulo 5 no se

han introducido nuevas ideas, sino que se proporcionan detalles concretos sobre como se pueden extender las aproximaciones previas y se presentan experimentos para poner a prueba dichas extensiones. Se consideran las MNs con forma de perceptrón multicapa y las extensiones a clasificación y regresión robusta de los métodos dispersos propuestos.

En resumen, esta tesis proporciona un conjunto de GPs dispersos que pueden ser usados para atacar problemas de aprendizaje supervisado a gran escala y que superan el actual estado del arte. Se ha intentado enfatizar la flexibilidad y extensibilidad de dichos modelos, de manera que puedan ser ajustados a las necesidades específicas de cada tarea concreta de manera directa.

Comparación de las metodologías propuestas

Los modelos discutidos en esta tesis se presentan de manera comparada en la Tabla 1. Cada una de las categorías (a las que pueden pertenecer uno o más modelos) está valorada de acuerdo a tres factores relevantes: Precisión (de las predicciones), resistencia al sobreajuste (en el sentido descrito en la Sección 2.5) y calidad de las varianzas predictivas (es decir, si evitan confiar excesivamente en sus propias predicciones). Como cabía esperar, ninguno de los modelos es el mejor en todos los aspectos. Sin embargo, dado el amplio rango de opciones disponibles, se puede presumir que, dada una tarea, al menos uno de ellos será apropiado.

Además, es posible agrupar estas categorías en dos tipos fundamentales:

(a) **Modelos que encajan en la estructura de una MN** (es decir, tienen la estructura de modelos lineales generalizados). Cuando los pesos de entrada se aprenden consiguen una elevada precisión, pero pueden sobreajustar si se usan directamente. Recurriendo a estrategias como la marginalización de fase en los SSGP, u otras más generales como usar el ruido acotado o la mezcla de redes del Capítulo 3, es posible reducir el impacto de este problema y mantener medias predictivas de alta calidad. La calidad de las varianzas predictivas es razonablemente buena.

(b) **Modelos que encajan en la estructura de los IDGPs** (es decir, asumen que todos los valores de la función latente son condicionalmente independientes dado un conjunto de variables, posiblemente pertenecientes a un dominio diferente). En

Método	Precisión	Resistencia al sobreaj.	Calidad de la var. estimada
SSGP-fijos, MN-fijos (MCN-fijos, MMLP-fijos)	media	muy alta	alta
SSGP	alta	media	media-baja
BN-MNs (BN-MCN, BN-MMLP)	alta	alta	baja
MNmix (MCNmix, MMLPmix)	muy alta	alta	media-alta
IDGPS (SPGP, FIFGP, TFIFGP)	media-alta	alta	alta

Tabla 1: Comparativa de las fortalezas y debilidades de los métodos propuestos en esta tesis. El significado de cada columna está descrito en el texto.

este caso, existen restricciones adicionales que aseguran que el IDGP se aproxima al GP completo lo mejor posible para el conjunto de pseudo-características seleccionadas (minimizando la divergencia de Kullback-Leibler, ver Sección 4.2). El apropiado manejo de las incertidumbres en estos modelos da lugar a unas varianzas predictivas de alta calidad. Sin embargo, precisamente por ser modelos más restringidos, normalmente no alcanzan el poder predictivo de las MNs. También son modelos más complejos, por lo que típicamente requerirán más tiempo de cómputo para un mismo número de funciones de base.

Líneas futuras

Algunas de las extensiones más relevantes a las ideas principales de esta tesis ya han sido expuestas en el Capítulo 5. Muchas otras, sin embargo, no se han incluido y serán objeto de futuras investigaciones. Algunas de ellas son:

- **Extensiones para manejar múltiples salidas y aprendizaje multitarea.** El concepto de "compartición de pesos" se utiliza en las redes neuronales —entre

otras cosas— para introducir dependencias entre diferentes salidas correspondientes a una misma entrada (regresión multi-salida), o entre diferentes tareas (regresión multi-tarea). Esta idea no podía ser usada en los GPs estándar, dado que en un GP los pesos han sido marginalizados y por lo tanto no están presentes, pero puede ser aplicada a muchos de los modelos desarrollados en esta tesis (aquellos que encajan en la estructura de una MN, mostrados en las primeras cuatro filas de la Tabla 6.1). Esto puede resultar útil para construir GPs dispersos que se beneficien de las mejoras de rendimiento proporcionadas por los métodos propuestos.

Dado que el aprendizaje multi-tarea trata con múltiples conjuntos de datos a la vez, su coste computacional es particularmente alto. Bonilla et al. (2008) recurren a la aproximación Nyström para acelerar los cálculos y mencionan otras alternativas a este efecto, tales como el SPGP. Algunas de nuestras propuestas, particularmente los IDGPs, podrían ser apropiados para esta tarea, aportando mayor precisión (en comparación con otros métodos dispersos). Además, los IDGPs permitirían nuevos tipos de acoplo entre tareas si se comparten las pseudo-características que los definen. Forzando a diferentes tareas a reutilizar las mismas pseudo-características, se podría conseguir que éstas requiriesen de un menor número de muestras por tarea para aprenderse con la misma precisión, incrementando así la capacidad predictiva.

- **Nuevas "funciones de extracción de características".** Cada IDGP está definido por una función de extracción de características. En esta tesis hemos considerado cuatro funciones de extracción, las cuales dan lugar al SPGP, FIFGP, TFIFGP y GPs multi-escala. Resultaría interesante explorar otras particularizaciones de los IDGPs en el futuro; en particular parece razonable que utilizar otros esquemas de enventanado (tales como múltiples ventanas siguiendo la distribución de los datos de entrada) podría resultar en métodos de mayor rendimiento.

- **Combinación con el esquema variacional de Titsias (2009).** Otra línea de trabajo prometedora consistiría en combinar los IDGPs con el método variacional de Titsias (2009) para obtener algoritmos de regresión dispersa que aproximen con mayor fidelidad la distribución del GP completo. El IDGP variacional se aproximaría al GP completo a medida que el número de pseudo-características aumentase, convergiendo a él en el límite.

- **Aplicaciones que requieran de inferencia entre dominios.** Hemos mencionado esta posibilidad en el Capítulo 4. Aunque actualmente no conocemos aplicaciones en las que se requiera hacer inferencia acerca de datos que provengan de diferentes dominios linealmente relacionados, tales aplicaciones pueden aparecer en el futuro. Por otra parte, incluso cuando todos los datos se encuentran en un mismo dominio, esta técnica puede ser útil para hacer inferencia sobre datos en otro (por ejemplo, inferir probabilísticamente cuál es la amplitud de un conjunto de componentes de frecuencia a partir de datos en el dominio del tiempo).

- **Combinación de las MNs con nuevas técnicas de regularización.** El sobreajuste es uno de los principales problemas que impiden el uso directo de las MNs. Aunque hemos proporcionado soluciones específicas para evitar el sobreajuste en esta tesis, podría ser interesante combinar las MNs con otras técnicas de regularización (existentes o futuras).

Contents

List of Figures

List of Tables

Symbols and notation

We use bold lower case for vectors and bold upper case for matrices. Subscript asterisk (*) is used to refer to test set quantities, either data points or latent variables. We use $p(\cdot)$ to denote both probabilities and probability densities.

$[\mathbf{A}]_{pq}$	Element (p, q) from matrix $\mathbf{A}$
$[\mathbf{a}]_p$	Element p from vector $\mathbf{a}$
$\mathbf{I}_q$	Identity matrix of size $q \times q$
$f(\cdot)$	Latent function, usually a Gaussian Process
$\mathcal{GP}(m(\cdot), k(\cdot, \cdot))$	Gaussian Process with mean $m(\cdot)$ and covariance function $k(\cdot, \cdot)$
$m(\mathbf{x})$	Mean function $m : \mathbb{R}^D \to \mathbb{R}$
$k(\mathbf{x}, \mathbf{x}')$	Covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, parametrized by the hyperparameters $\boldsymbol{\theta}$
$\boldsymbol{\theta}$	Hyperparameters of the covariance function
n	Number of data points in the training set
n_*	Number of data points in the test set
j	Running index over data points
m	Number of basis functions
i	Running index over basis functions
h	Number of spectral points

r	Running index over spectral points
D	Dimension of the input space, or equivalently, number of features in the data set
$(\mathbf{x}, y)$	Training data point within the training set, consisting of input vector $\mathbf{x} \in \mathbb{R}^D$ and its corresponding scalar output (label) y
$(\mathbf{x}_*, y_*)$	Test data point consisting of input vector $\mathbf{x}_* \in \mathbb{R}^D$ and its corresponding scalar output (label) y_*
$\mathcal{D}$	Data set consisting of n data points $\{\mathbf{x}_j, y_j\}_{j=1}^n$
$\mathbf{X}$	Matrix of size $n \times D$ containing all input vectors from the training set, $[\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$
$\mathbf{y}$	Vector of size $n \times 1$ containing observed values (targets, labels) from the training set, $[y_1 \ldots y_n]^\top$
$\mathbf{f}$	Vector of size $n \times 1$, containing the latent function evaluated at the training points $[f(\mathbf{x}_1) \ldots f(\mathbf{x}_n)]^\top$
$p(\mathbf{y}\|\mathbf{X})$	Multivariate probability distribution of vector $\mathbf{y}$, given matrix $\mathbf{X}$
$\mathbb{E}[\cdot]$	Expectation of a random variable
$\mathbb{V}[\cdot]$	Variance of a random variable
$\mathrm{cov}(\cdot, \cdot)$	Covariance of two random variables
$\mathcal{N}(\mathbf{f}\|\mathbf{m}, \mathbf{K})$	Multivariate normal distribution over $\mathbf{f}$ with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{K}$, check (B.1) for the complete expression
$\mathbf{K}_{\mathbf{ab}}$	Matrix of size $\mathrm{length}(\mathbf{a}) \times \mathrm{length}(\mathbf{b})$ containing the prior covariance of vectors $\mathbf{a}$ and $\mathbf{b}$, with elements $[\mathbf{K}_{\mathbf{ab}}]_{pq} = \mathrm{cov}([\mathbf{a}]_p, [\mathbf{b}]_q)$
$\mathbf{k}_{\mathbf{f}*}$	Vector of size $n \times 1$ containing the prior covariance of test point $\mathbf{x}_*$ and the training inputs $\{\mathbf{x}_j\}_{j=1}^m$, with elements $[\mathbf{k}_{\mathbf{f}*}]_j = \mathrm{cov}(\mathbf{x}_j, \mathbf{x}_*)$
k_{**}	Autocovariance $\mathrm{cov}(\mathbf{x}_*, \mathbf{x}_*)$ at test point $\mathbf{x}_*$
$\mathbf{k}_{\mathbf{a}}(\mathbf{x})$	Vector function $\mathbf{k}_{\mathbf{a}} : \mathbb{R}^D \to \mathbb{R}^{\mathrm{length}(\mathbf{a})}$, with elements $[\mathbf{k}_{\mathbf{a}}(\mathbf{x})]_p = \mathrm{cov}([\mathbf{a}]_p, \mathbf{x})$

Chapter 1

Introduction

Machine learning tasks can be broadly divided into three main categories, namely supervised learning, unsupervised learning and reinforcement learning. Some tasks, such as semi-supervised learning, lie in between these areas. In this thesis we will deal with supervised learning on large-scale problems. Since supervised learning can be used as a building block for other learning tasks, the ideas developed here can also find application within other machine learning categories.

Gaussian Processes (GPs) have been shown to provide state-of-the-art performance in supervised learning tasks such as regression —Rasmussen (1996)— or classification —Naish-Guzman and Holden (2008)—. In addition to being highly accurate, they also present a number of other appealing features: Probabilistic predictions, no overfitting, a simple model selection scheme, etc. Unfortunately, GPs are not appropriate to handle large data sets directly, due to their high computational cost. In this thesis we will develop and test new sparse GP models that strive to reach the quality of full GPs while dramatically reducing computation time. We will show how the state-of-the-art sparse GP model is often outperformed by the novel methods, as well as discuss the strengths and weaknesses of our proposals.

This chapter is organized as follows: In Section 1.1 standard GPs for supervised learning are reviewed; in Section 1.2 previous relevant contributions to the development of sparse GP models are summarized; and in Section 1.3 we outline the contents of the rest of the thesis.

1.1 Gaussian Processes (GPs)

The first use of Gaussian Processes (GPs) for multivariate regression dates back to Matheron (1973) under the name of *kriging*, within the Geostatistics community. This name is due to the mining engineer and Geostatistics pioneer Daniel G. Krige. Earlier uses of GPs as time domain stochastic processes appear in classical texts such as Wiener (1949).

GPs for regression were first introduced to the Statistics community by the seminal paper of O'Hagan (1978) and then to the Machine Learning community by Williams and Rasmussen (1996). They have since been actively developed, and extensions for classification, robust regression, dimensionality reduction and other core machine learning tasks have appeared. For a thorough treatment of GPs for machine learning, see Rasmussen and Williams (2006).

1.1.1 What is a GP?

A stochastic process $f(\mathbf{x})$ with $f : \mathbb{R}^D \to \mathbb{R}$ is a GP if and only if any finite collection of its samples $\mathbf{f} = [f(\mathbf{x}_1) \dots f(\mathbf{x}_n)]^\top$ forms a multivariate Gaussian random variable.

A GP $f(\mathbf{x})$ is completely specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$. Since we will consider only real-valued GPs, these are defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned} \tag{1.1}$$

We will use the following notation to compactly define $f(\mathbf{x})$ as a GP:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')) \ .$$

Usually we will work with zero-mean models, such that $m(\mathbf{x}) = 0$. From the definition it is also clear that the covariance function is symmetric with respect to the argument order.

In the traditional setting of stochastic processes defined over time, $\mathbf{x}$ would be reduced to a (possibly discrete) scalar variable. Then $f(\mathbf{x})$ would represent the random variable corresponding to time point $\mathbf{x}$. In the more general setting presented here, the

GP is defined over some input space $\mathbb{R}^D$ (the index set), so that there is a random variable assigned to every input space point.

Using the above definitions, the joint distribution of a set of random variables $\mathbf{f} = [f(\mathbf{x}_1) \dots f(\mathbf{x}_n)]^\top$ is[1]:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m},\ \mathbf{K}), \tag{1.2}$$

where $\mathbf{m} = [m(\mathbf{x}_1) \dots m(\mathbf{x}_n)]^\top$ and $[\mathbf{K}]_{pq} = k(\mathbf{x}_p, \mathbf{x}_q)$.

The above definition uses $k(\cdot, \cdot)$ to individually specify the elements of the covariance matrix. This ensures that the GP is *consistent*, i.e., it fulfills the marginalization property. Applying (1.2) to some set of samples $\mathbf{f}$ yields the same joint distribution as applying it to a superset of $\mathbf{f}$ and then marginalizing out the additional samples. Note that other possibilities, such as using some function to specify the entries of the inverse covariance matrix would not fulfill the marginalization property and therefore would not yield a valid GP.

1.1.2 Covariance functions

Covariance functions encode information about the smoothness and overall properties of the underlying GP. According to their definition (1.1), they provide a measure of the degree of correlation between any pair of samples of a GP as a function of the location of those samples in the input domain. Covariance functions specify our beliefs about how two samples are related. To allow for some degrees of freedom in the definition of this relation, additional dependence on some parameters (usually, scaling factors) can be introduced. These parameters are referred to as covariance *hyperparameters*, since they do not parameterize the model itself, but its prior statistics.

The sufficient and necessary condition for any function to be a valid covariance function is it being positive semidefinite. This ensures that it will produce valid (positive semidefinite) covariance matrices when evaluated on a finite set of data. Usually, we will also want it to produce higher covariance values for samples corresponding to points which are closer in the input domain.

Of special interest is the class of covariance functions that only depend on the difference of their arguments. Such covariance functions are called *stationary*, and

[1] This is standard notation for a Gaussian distribution over $\mathbf{f}$ with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{K}$, check (B.1) for the complete expression.

fulfill that:

$$k_{\text{st}}(\mathbf{x},\ \mathbf{x}') \ = \ k_{\text{st}}(\mathbf{x} + \Delta\mathbf{x},\ \mathbf{x}' + \Delta\mathbf{x}), \quad \forall \Delta\mathbf{x} \in \mathbb{R}^D. \tag{1.3}$$

Such covariance functions are often expressed in terms of this difference:

$$k_{\text{st}}(\mathbf{x}' - \mathbf{x}) \ = \ k_{\text{st}}(\boldsymbol{\tau}). \tag{1.4}$$

This class of covariance functions produce GPs with constant pointwise variance $k(\mathbf{0})$ at every point. Coupled with a constant mean function (such as the usual $m(\mathbf{x}) = 0$), they can be useful to model functions which are known to have a stationary behavior, and thus exhibit constant power.

The most common choice by far for the covariance function is the anisotropic, unnormalized Gaussian:

$$k_{\text{ARD SE}}(\mathbf{x}, \mathbf{x}') \ = \ \sigma_0^2 \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\ell_d^2}\right], \tag{1.5}$$

where x_d is component d of vector $\mathbf{x}$, and $\sigma_0^2 and \{\ell_d\}_{d=1}^D$ are hyperparameters. It is also called Automatic Relevance Determination Squared Exponential (ARD SE) for reasons that we will see shortly.

Since this covariance function meets (1.3), it is stationary. The pointwise variance of the corresponding GP at any point $\mathbf{x}$ is $k(\mathbf{x}, \mathbf{x}) = k(\mathbf{0}) = \sigma_0^2$. We will therefore call σ_0^2 the "signal power hyperparameter", assuming $m(\mathbf{x}) = 0$. The remaining hyperparameters $\{\ell_d\}_{d=1}^D$ are called length-scales, because they have the effect of scaling each input dimension. They can be used to control how rapidly the covariance between two points decays. As the length-scale corresponding to some input dimension grows, the effect of that input dimension in the covariance is reduced and, in the infinite limit, disappears completely. When (1.5) is coupled with a model selection scheme (which automatically selects the most appropriate hyperparameters, see Section 1.1.5), this covariance function can effectively remove irrelevant input dimensions and hence the "Automatic Relevance Determination" part of its name.

Fig. 1.1 shows the shape of the ARD SE covariance function. It decays smoothly to zero and its width is controlled by the corresponding length-scale. We will be mainly concerned with this covariance function until Section 5.1, where the ARD Multi-Layer Perceptron (ARD MLP) covariance function is presented.

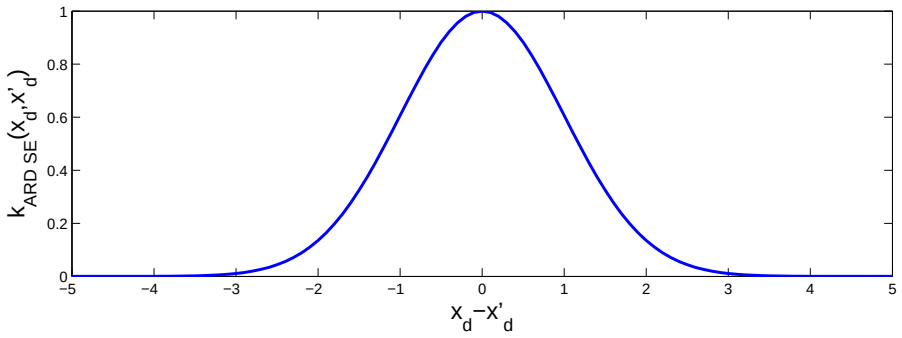


Figure 1.1: One-dimensional ARD SE covariance function (1.5).

1.1.3 Regression using GPs

The regression problem can be stated as follows: Given a data set $\mathcal{D}$ consisting of n D-dimensional input vectors $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]$ and a corresponding set of continuous scalar outputs $\mathbf{y} = [y_1 \dots y_n]^\top$, predict the output y_* corresponding to some new test input $\mathbf{x}_*$. We refer to $\mathcal{D} \equiv \{\mathbf{X}, \mathbf{y}\}$ as the "training set", since it contains all the information that is available to determine the input-output mapping that we are seeking.

Traditional approaches to this problem are based on proposing some input-output mapping (such as multi-layer perceptrons, radial basis functions networks, etc.) and then select the free parameters of this mapping (the network weights) so as to minimize some cost functional (typically, the squared error over the training set). This mapping provides a point estimate $\hat{y}_*$ of the output at any test point $\mathbf{x}_*$.

Regression with GPs, on the other hand, is performed in a Bayesian way. This means that the parameters of the model are integrated out and, instead of point estimates, full posterior distributions are provided for the parameters of the model and the predicted outputs.

The process of Bayesian inference can be summarized as follows:

1. Some parametric model is proposed, such that the probability of the observed data given a concrete set of parameters can be computed (*likelihood*).
2. Some a priori probability is defined over the parameters space (*prior*).
3. From the above definitions, a posterior distribution over parameters is obtained (*posterior* over parameters).

4. Forming the product of the likelihood and the posterior and integrating over the parameters space, predictive distributions for new observations can be obtained.

The posterior distribution at step 3 is obtained from the likelihood and the prior using Bayes theorem, hence the name "Bayesian". We will now follow these steps for the regression task.

1.1.3.1 Likelihood

The GP regression model assumes that the outputs can be modeled as some noiseless latent function $f(\mathbf{x})$ of the inputs plus independent noise:

$$y \;=\; f(\mathbf{x}) + \varepsilon. \tag{1.6}$$

In this case, the latent function plays the role of the model parametrization. Assuming ε is zero-mean Gaussian noise of power σ^2, we can write the likelihood of any observation as

$$p(y|f(\mathbf{x})) = \mathcal{N}(y|f(\mathbf{x}),\ \sigma^2). \tag{1.7}$$

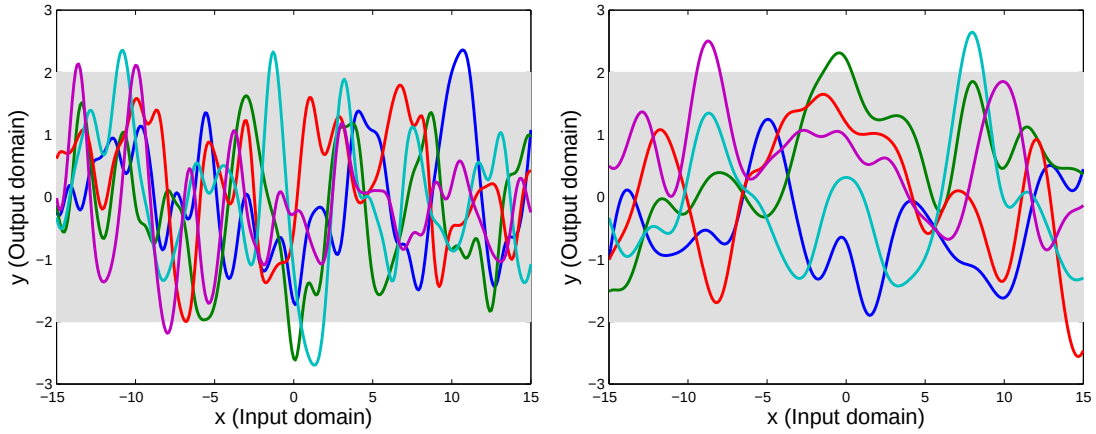
1.1.3.2 Prior

Now we need a distribution (over functions) to define the prior probability of $f(\mathbf{x})$, and here is where GPs come in handy. We set the following zero mean[2] prior:

$$f(\mathbf{x}) \;\sim\; \mathcal{GP}(0,\ k(\mathbf{x}, \mathbf{x}')). \tag{1.8}$$

Fig. 1.2 shows 5 samples (functions) drawn from this prior distribution. The ARD SE covariance function with $\sigma_0^2 = 1$ is used. Each panel was created with a different length-scale to illustrate the effect of this hyperparameter on the prior. The smaller the length-scale is, the faster the latent function is expected to fluctuate. The shaded region gives an idea of the pointwise distribution by including twice the standard deviation around the mean.

[2]It is customary to subtract the sample mean from data, and then to assume a zero mean model. Extensions to handle arbitrary mean data are straightforward.



(a) Fast length-scale, $\ell^2 = 0.5$

(b) Slow length-scale, $\ell^2 = 2$

Figure 1.2: Samples drawn from the prior distribution of the latent function. The shaded area accounts for 95% of the prior probability.

1.1.3.3 Posterior over the latent function

We are now ready to obtain the posterior distribution of the latent function at any location. We will compute it at the training points $\mathbf{X}$ and an arbitrary number n_* of test points $\mathbf{X}_* = \{\mathbf{x}_{*i}\}_{i=1}^{n_*}$ using Bayes rule:

$$p(\mathbf{f}, \mathbf{f}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*) = \frac{p(\mathbf{y}|\mathbf{f}, \mathbf{f}_*, \mathbf{X}, \mathbf{X}_*)p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*)}{p(\mathbf{y}|\mathbf{X}, \mathbf{X}_*)} = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*)}{p(\mathbf{y}|\mathbf{X})}, \quad (1.9)$$

where we have defined $\mathbf{f} = [f(\mathbf{x}_1) \dots f(\mathbf{x}_n)]^\top$ and $\mathbf{f}_* = [f(\mathbf{x}_{*1}) \dots f(\mathbf{x}_{*n_*})]^\top$. The second identity follows from the (conditional) independence relations implied by the chosen likelihood (1.7).

Likelihood (1.7) and prior (1.8), when evaluated at the relevant locations, yield:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f},\ \sigma^2\mathbf{I}_n) \quad (1.10)$$

$$p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix}\middle|\ \mathbf{0},\ \begin{bmatrix}\mathbf{K_{ff}} & \mathbf{K_{f*}}\\ \mathbf{K_{f*}}^\top & \mathbf{K}_{**}\end{bmatrix}\right), \quad (1.11)$$

where the elements of the covariance matrices[3] are $[\mathbf{K_{ff}}]_{pq} = k(\mathbf{x}_p, \mathbf{x}_q)$, $[\mathbf{K_{f*}}]_{pq} = k(\mathbf{x}_p, \mathbf{x}_{*q})$, $[\mathbf{K}_{**}]_{pq} = k(\mathbf{x}_{*p}, \mathbf{x}_{*q})$, and $\mathbf{I}_n$ is the identity matrix of size n.

[3]When a single test point is considered, some of these covariance matrices become vectors or scalars, and we will vary capitalization and/or bold type accordingly.

The numerator of (1.9) is then the product of (1.10) and (1.11), which is straightforward to compute and provides an intuitive view of the relationship among the involved variables:

$$\begin{aligned} p(\mathbf{y}|\mathbf{f})p(\mathbf{f},\mathbf{f}_*|\mathbf{X},\mathbf{X}_*) &= p(\mathbf{f},\mathbf{f}_*,\mathbf{y}|\mathbf{X},\mathbf{X}_*) \\ &= \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\\ \mathbf{y}\end{bmatrix}\middle|\, \mathbf{0},\, \begin{bmatrix}\mathbf{K_{ff}} & \mathbf{K_{f*}} & \mathbf{K_{ff}}\\ \mathbf{K_{*f}} & \mathbf{K_{**}} & \mathbf{K_{*f}}\\ \mathbf{K_{ff}} & \mathbf{K_{f*}} & \mathbf{K_{ff}}+\sigma^2\mathbf{I}_n\end{bmatrix}\right), \end{aligned} \tag{1.12}$$

Finally, we can compute the posterior distribution over the latent values $\mathbf{f}$ and $\mathbf{f}_*$. From (1.12) and using (B.3) to condition on $\mathbf{y}$, we obtain the explicit posterior at any location:

$$\begin{aligned} p(\mathbf{f},\mathbf{f}_*|\mathbf{y},\mathbf{X},\mathbf{X}_*) = \mathcal{N}\Bigg(&\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix}\middle|\, \begin{bmatrix}\mathbf{K_{ff}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{y}\\ \mathbf{K_{*f}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{y}\end{bmatrix},\\ &\begin{bmatrix}\mathbf{K_{ff}}-\mathbf{K_{ff}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{K_{ff}} & \mathbf{K_{f*}}-\mathbf{K_{ff}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{K_{f*}}\\ \mathbf{K_{*f}}-\mathbf{K_{*f}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{K_{ff}} & \mathbf{K_{**}}-\mathbf{K_{*f}}(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{K_{f*}}\end{bmatrix}\Bigg). \end{aligned} \tag{1.13}$$

Marginalizing out $\mathbf{f}$ from (1.13) using (B.2), we have the posterior distribution of the latent function $f_* = f(\mathbf{x}_*)$ at any point:

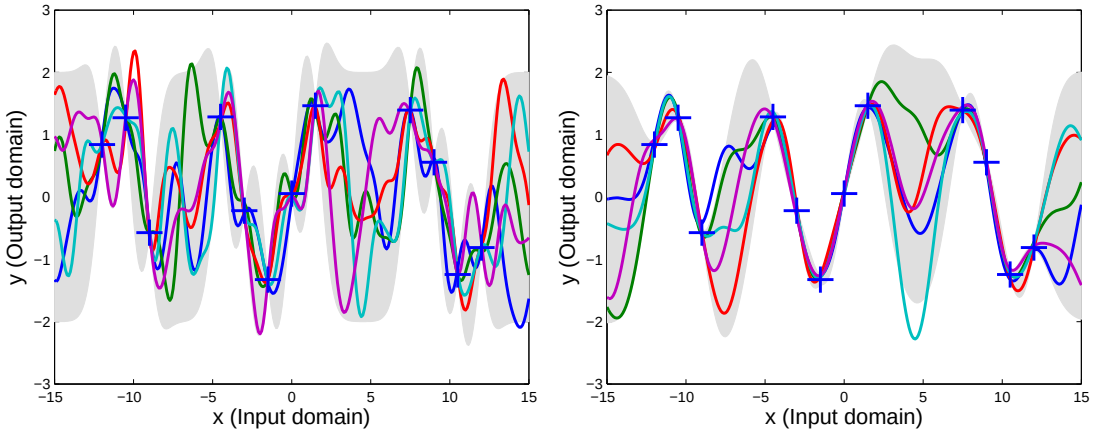
$$p(f_*|\mathbf{y},\mathbf{X},\mathbf{x}_*) = \mathcal{N}(f_*|\mu(\mathbf{x}_*),\sigma^2(\mathbf{x}_*)) \tag{1.14a}$$

$$\mu(\mathbf{x}_*) = \mathbf{k_f}(\mathbf{x}_*)^\top(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{y} \tag{1.14b}$$

$$\sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k_f}(\mathbf{x}_*)^\top(\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{k_f}(\mathbf{x}_*)\,, \tag{1.14c}$$

where $\mathbf{k_f}(\mathbf{x}_*)$ is a vector-valued function that computes the covariance between the set of values given as subindex and the latent function evaluated at the point given as argument.

As an example of posterior distribution for the latent function, consider Fig. 1.3. It shows 12 randomly spaced observations coming from a low noise sine, and 5 samples (functions) drawn from the resulting posterior distribution. The shaded region is centered on the posterior mean and denotes twice the square root of the posterior variance around it. Both panels show reasonable predictions, but the right panel, whose length-scale is better suited to the input data, produces more accurate and confident predictions, and is visually more appealing and less wiggly. Hyperparameters can be automatically selected to best fit the input data, as we shall see in Section 1.1.5.



(a) Fast length-scale, $\ell^2 = 0.5$

(b) Slow length-scale, $\ell^2 = 2$

Figure 1.3: Samples drawn from the posterior distribution of the latent function. The shaded area accounts for 95% of the posterior probability. Crosses represent training data.

1.1.3.4 Posterior over the outputs

The predictive distribution for some new output y_* follows from (1.7) and (1.14):

$$p(y_*|\mathbf{y}, \mathbf{X}, \mathbf{x}_*) = \int p(y_*|f_*)p(f_*|x_*, \mathbf{X}, \mathbf{y})df_* = \mathcal{N}(y_*|\mu_*, \sigma_*^2) \tag{1.15a}$$

$$\mu_{\text{GP}*} = \mathbf{k}_{\mathbf{f}*}^\top(\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y} \tag{1.15b}$$

$$\sigma^2_{\text{GP}*} = k_{**} + \sigma^2 - \mathbf{k}_{\mathbf{f}*}^\top(\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{k}_{\mathbf{f}*} \ , \tag{1.15c}$$

where $\mathbf{k}_{\mathbf{f}*} = \mathbf{k}_{\mathbf{f}}(\mathbf{x}_*) = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)]^\top$ and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$.

The noise term $\sigma^2\mathbf{I}_n$ that gets added to $\mathbf{K}_{\mathbf{ff}}$ (which in turn is positive semidefinite by definition) has a regularizing effect and ensures proper conditioning before inversion. Additional accuracy and numerical stability can be achieved if these computations are performed using Cholesky factorizations, see Section D.1.

In the simple regression case, the only difference between the posterior of the latent function (1.14) and the posterior of the outputs (1.15) is an excess σ^2 variance, since the latter results from adding white noise to the former (1.6). This close similarity does not hold in general (see for instance the classification model in Section 1.1.4).

1.1.3.5 Computation and storage costs

The cost of computing equations (1.15b) and (1.15c) for a single test sample is dominated by the inversion of matrix $(\mathbf{K_{ff}} + \sigma^2\mathbf{I}_n)$, which is $\mathcal{O}(n^3)$. When predictions for several test cases are needed, we can reuse previous computations, so that computing (1.15b) and (1.15c) takes an extra $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ time per new sample, respectively.

Storage cost is dominated by the need to keep the covariance matrix and its inverse in memory, and scales as $\mathcal{O}(n^2)$. Of course, it is possible to trade computation time for storage space and recompute their elements each time they are needed.

This high computational complexity means that, in practice, modern desktop computers can only handle GP regression for relatively small data sets, usually up to a few thousand samples, see Rasmussen (2003).

1.1.4 Robust regression and classification

The robust regression task coincides with that of plain regression discussed above, but tries to reduce the sensitivity of the model to outliers in the training data set.

A simple approach to robust regression with GPs is to place a leptokurtic[4] prior on noise ε, which in turn translates into some non-Gaussian likelihood. The prior on the latent function remains unchanged.

The classification problem can be stated exactly as the regression problem, but constraining the outputs to take only discrete values. In the common case of binary classification, i.e. $y \in \{+1, -1\}$, the following likelihood can be used:

$$p(y = +1|f(\mathbf{x})) = s(f(\mathbf{x}))$$

where $s(\cdot)$ is any sigmoid function (such as the logistic or probit). If the symmetry condition $s(-z) = 1 - s(z)$ is fulfilled, we have that

$$p(y = -1|f(\mathbf{x})) = 1 - p(y = +1|f(\mathbf{x})) = s(-f(\mathbf{x}))$$

and we can express the likelihood of any sample compactly as $p(y|f(\mathbf{x})) = s(yf(\mathbf{x}))$. As with robust regression, the same GP prior (1.8) can be used on latent function $f(\cdot)$. Further details on GP classification are given in Section 5.4.1.

[4] A distribution with positive excess kurtosis (i.e., a higher kurtosis value than a Gaussian distribution) is called leptokurtic. Leptokurtosis implies a sharp central peak and fat tails.

Therefore, both robust regression and binary classification models result from a slight modification of the regression model. However, this small modification is of major importance, since having a non-Gaussian likelihood means we cannot obtain an analytical expression of the posterior any longer. Inference in such models is possible only by resorting to approximate techniques.

We will develop all the sparse GP models introduced in this thesis for the simpler, analytically tractable case of regression, and then generalize them for robust regression and binary classification in Section 5.2.

1.1.5 Model selection

In order to turn the models mentioned so far into usable algorithms, we need an strategy to select any free hyperparameters. These include covariance hyperparameters (such as σ_0^2 and $\{\ell_d\}_{d=1}^D$ for the ARD SE case) and the noise power hyperparameter σ^2. We will collectively refer to them as $\boldsymbol{\theta}$. The dependence on $\boldsymbol{\theta}$ is explicitly stated in this section, but it will generally be omitted for the sake of clarity.

Following a proper Bayesian treatment, one should place a prior on the hyperparameters and then integrate them out. However, pursuing this option involves solving several possibly intractable integrals, and therefore resorting to computationally expensive approximations. For this reason, this approach is often avoided.

A simpler and common approach is to use a single point estimate of the hyperparameters, the one that maximizes the marginal likelihood of the model (also known as *evidence*). This approach is known as type II Maximum Likelihood (ML-II).

The evidence can be obtained from the likelihood and the prior, marginalizing out the latent function:

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f},\mathbf{X},\boldsymbol{\theta})p(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})d\mathbf{f}. \tag{1.16}$$

For the specific case of regression, applying the marginalization property to (1.12) and explicitly including the dependence on $\boldsymbol{\theta}$, we have

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}|\mathbf{0},\ \mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I}_n). \tag{1.17}$$

Since we are only interested in finding the maximum wrt $\boldsymbol{\theta}$, often the equivalent problem of minimizing the Negative Log-Marginal Likelihood (NLML) is solved,

which is numerically better behaved (since $p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta})$ tends to zero for high n). The NLML is:

$$-\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = +\frac{1}{2}\mathbf{y}^\top \left(\mathbf{K_{ff}}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{y} + \frac{1}{2}|\mathbf{K_{ff}}+\sigma^2\mathbf{I_n}| + \frac{n}{2}\log(2\pi), \quad (1.18)$$

and can be computed in $\mathcal{O}(n^3)$ time. In practice, this equation should be implemented using Cholesky factorizations as described in Section D.1. If analytical expressions for the gradients are available (which is usually the case), a local minimum can be found using (conjugate) gradient descent. A single step of this process takes roughly the same time as computing (1.15b) and (1.15c) for n test samples, and usually several steps are needed to reach a solution. Therefore, the time consumed by model selection often dominates the overall process of learning.

Another popular model selection procedure is cross-validation. While cross-validation is often the only possibility for model selection on non-Bayesian approaches, it limits the parameter space to a discrete grid, which has to be thoroughly explored and whose size grows exponentially with the number of free parameters. In contrast, Bayesian methods such as GPs can benefit from ML-II continuous parameter space exploration, which usually takes a modest number of iterations.

Additionally, the inclusion of Occam's razor is automatic in GPs. While the first term in equation (1.18) measures data fit, the second term penalizes model complexity. These terms express the (opposing) desiderata of both fitting training data accurately and avoiding overcomplex models that could perform badly on out-of-sample data. Achieving a good trade off between these two purposes has been a long-standing problem for non-Bayesian models, but one which is automatically handled by GPs.

1.2 Summary of previous sparse GP approximations

Several sparse approximations have been proposed in order to make GP regression affordable when working with large data sets, e.g.: Csató and Opper (2002); Seeger et al. (2003); Silverman (1985); Smola and Bartlett (2001); Snelson and Ghahramani (2006); Tresp (2000); Williams and Seeger (2001); etc. We will now review the main milestones in the development of sparse GPs, devoting special attention to the current state of the art, the Sparse Pseudo-inputs GP (SPGP) on Section 1.2.5. SPGP is used as

a benchmark in this thesis and forms the basis of the sparse inference method presented in Chapter 4.

Many sparse GP approximations were initially thought of as approximate inference methods for GP models with some exact prior. In Quiñonero-Candela and Rasmussen (2005) it is shown that, for most proper probabilistic approximations, it is possible to cast the approximate GP as an exact GP with an approximate prior. Using this alternative and perhaps more natural interpretation, the cited work unifies different approximations within a single framework, in spite of their varying origins and motivations. The new interpretation provides more hints about the quality of each approximation, whether it corresponds to an exact probabilistic model or not and how all of them relate to each other. In the subsections below, for each sparse GP approximation we also note the corresponding *effective prior* within the unifying framework described in Quiñonero-Candela and Rasmussen (2005).

1.2.1 Subset of data

Probably the simplest way to reduce complexity when faced with a big data set is just to discard the data that we cannot afford to process. This is a reasonable option when data is redundant enough, so that most of the available information can be present in some selected subset.

If all data points are equally informative, this subset can be selected randomly, but there is usually an advantage to more sophisticated selection heuristics. Several greedy selection criteria have been proposed, such as the Differential Entropy Score, Lawrence et al. (2003), and Information Gain, Seeger et al. (2003). In both cases, samples added to the subset are chosen to maximize some measure of the information held in it. After the inclusion of a new sample, relevant matrices are incrementally grown using low rank updates, so that sample selection does not increase the overall computational complexity of sparse regression.

1.2.2 The Nyström method

Using the Nyström method as described in Baker (1977), an approximation to the eigenvectors and eigenvalues of a matrix can be obtained. In Williams and Seeger

(2001) this idea is applied for efficient GP regression: A few columns of the covariance matrix are used to approximately obtain its eigenvectors and eigenvalues, which are in turn used to reconstruct the full matrix. This yields the following low rank approximation to the covariance matrix:

$$\mathbf{K_{ff}} \approx \mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{fu}^{\top}}, \tag{1.19}$$

where $\mathbf{u}$ a re the so-called *inducing variables*. For the Nyström approximation $\mathbf{u}$ is a subset of latent variables $\mathbf{f}$ of size $m \ll n$. Set of inputs $\overline{\mathbf{X}} \subset \mathbf{X}$ corresponding to $\mathbf{u}$ is called the *active set*. Recall that we use $\mathbf{K_{ab}}$ to refer to the covariance matrix between the elements from $\mathbf{a}$ and $\mathbf{b}$. Therefore, $\mathbf{K_{fu}}$ is an $n \times m$ matrix consisting of a subset of the columns of $\mathbf{K_{ff}}$ and $\mathbf{K_{uu}}$ is an $m \times m$ matrix consisting of the same subset of the rows and columns of $\mathbf{K_{ff}}$.

The Nyström method can be used to approximate the prior covariance between any two sets of variables, so we will also use the more general definition $\mathbf{Q_{ab}} = \mathbf{K_{au}}\mathbf{K_{uu}^{-1}}\mathbf{K_{bu}^{\top}}$.

Replacing $\mathbf{K_{ff}}$ with low rank matrix $\mathbf{Q_{ff}}$ on equations (1.15b) and (1.15c), matrix inversion lemma (A.1) can be applied. With this approximation, precomputations can be performed in $\mathcal{O}(m^2 n)$ time and predictions for new test samples can be made in $\mathcal{O}(n)$ time for the mean and $\mathcal{O}(nm)$ for the variance. Storage needs are reduced to $\mathcal{O}(nm)$, since the full covariance matrix is never used.

It is worth noting that this is a numerical approximation with no probabilistic foundation and can produce meaningless results. The effective joint prior for training and test data is

$$p_{\text{Nyst}}(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \middle| \, \mathbf{0}, \begin{bmatrix} \mathbf{Q_{ff}} & \mathbf{K_{f*}} \\ \mathbf{K_{*f}} & \mathbf{K_{**}} \end{bmatrix} \right), \tag{1.20}$$

where the exact prior covariance for training data has been replaced with the Nyström approximation, whereas the remaining prior covariances are kept exact. Doing so yields an inconsistent joint probability distribution whose prior covariance is not even guaranteed to be positive definite. The approximate posterior cannot be regarded as GP and absurd results such as a negative predicted variances can occur. The approximation can be too crude for low m, but as m approaches n, it trivially converges to the full GP.

1.2.3 Subset of regressors

This approximation stems from considering a degenerate GP model that yields the same predictive mean as the standard GP model described in Section 1.1.3. It was introduced by Silverman (1985). The degenerate model is a generalized linear regression model of the form:

$$f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}, \mathbf{x}_j)$$

with Gaussian prior on the weights $p(\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\alpha}|\mathbf{0}, \mathbf{K}_{\mathbf{ff}}^{-1})$. It is easy to show that the predictive mean of this model is that of a standard GP (1.15b). The predictive variance, however, is different, as we will see below.

Sparsity (and therefore computational advantages) appears if we only consider a subset of the regressors (corresponding to the active set):

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \quad \text{with} \quad p(\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\alpha}|\mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{-1}).$$

This subset of regressors (SR) model is equivalent to the following prior on $f(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(0,\ k_{\text{SR}}(\mathbf{x}, \mathbf{x}')) \quad \text{with} \quad k_{\text{SR}}(\mathbf{x}, \mathbf{x}') = \mathbf{k}_{\mathbf{u}}(\mathbf{x})^\top \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{k}_{\mathbf{u}}(\mathbf{x}'), \tag{1.21}$$

where $\mathbf{k}_{\mathbf{u}}(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}) \dots k(\mathbf{x}_m, \mathbf{x})]^\top$. Note that this approximate covariance function, when evaluated for every input pair, produces the Nyström approximation (1.19). The difference between the Nyström method and SR is that the former only replaces the training data covariance matrix, whereas the latter replaces the covariance function itself, thus yielding a proper GP.

The posterior mean and variance of the outputs under SR are:

$$\begin{aligned} p_{\text{SR}}(y_*|x_*, \mathcal{D}) &= \mathcal{N}(y_*|\mu_{\text{SR}*}, \sigma^2_{\text{SR}*}) \\ \mu_{\text{SR}*} &= \mathbf{k}_{\mathbf{u}*}^\top (\mathbf{K}_{\mathbf{fu}}^\top \mathbf{K}_{\mathbf{fu}} + \sigma^2 \mathbf{K}_{\mathbf{uu}})^{-1} \mathbf{K}_{\mathbf{fu}}^\top \mathbf{y} \\ \sigma^2_{\text{SR}*} &= \sigma^2 + \sigma^2 \mathbf{k}_{\mathbf{u}*}^\top (\mathbf{K}_{\mathbf{fu}}^\top \mathbf{K}_{\mathbf{fu}} + \sigma^2 \mathbf{K}_{\mathbf{uu}})^{-1} \mathbf{k}_{\mathbf{u}*}\,, \end{aligned}$$

which can be derived from (1.15) using the approximate covariance function (1.21), and we use the shorthand $\mathbf{k}_{\mathbf{u}*} = \mathbf{k}_{\mathbf{u}}(\mathbf{x}_*)$. As mentioned before, if instead of a subset we used all regressors (by setting $\mathbf{u} = \mathbf{f}$), this predictive mean would match that of the full GP (1.15b), but the predictive variance would be different, generally an underestimation of the correct value.

This problem gets worse when only a subset of the regressors is used. For instance, when approximating a stationary local covariance function, such as the often-used ARD SE (1.5), the prior pointwise variance $k_{\text{SR}}(\mathbf{x}, \mathbf{x})$ tends to zero as $\mathbf{x}$ moves away from the active set (from the definition (1.21) and the local character of $\mathbf{k}_{\mathbf{u}}(\mathbf{x})$). Since the posterior variance is never bigger than the prior variance, the predicted uncertainty of the latent function will tend to zero as we move away from training data, precisely where maximum uncertainty is expected.

The SR approximation does not specify a concrete method to select the active set and the simplest approach is just to select a random subset of the input data. A more refined option is to use the greedy forward selection method presented in Smola and Bartlett (2001).

Following Quiñonero-Candela and Rasmussen (2005), this approximation can be summarized by the effective joint prior on $\mathbf{f}, \mathbf{f}_*$, which tries to approximate the original prior (1.11) while producing computationally simpler posterior expressions:

$$p_{\text{SR}}(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \middle| \, \mathbf{0}, \, \begin{bmatrix} \mathbf{Q}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{f}*} \\ \mathbf{Q}_{*\mathbf{f}} & \mathbf{Q}_{**} \end{bmatrix} \right). \tag{1.22}$$

Precomputations can be performed in $\mathcal{O}(m^2 n)$ time and predictions for new test samples can be made in $\mathcal{O}(m^2)$ time for the mean and $\mathcal{O}(m^2)$ for the variance. Storage needs are reduced to $\mathcal{O}(nm)$, as with the Nyström approximation.

1.2.4 Projected Latent Variables

Projected Latent Variables (PLV) were introduced in Seeger et al. (2003), taking ideas from Csató and Opper (2002). This method is referred to as Projected Process Approximation (PPA) in Rasmussen and Williams (2006) and Deterministic Training Conditional (DTC) in Quiñonero-Candela and Rasmussen (2005).

PLV's contribution with respect to SR is to replace the approximate prior on the *test* latent variables with the exact one. Following Quiñonero-Candela and Rasmussen (2005), this method can be summarized as:

$$p_{\text{PLV}}(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \middle| \, \mathbf{0}, \, \begin{bmatrix} \mathbf{Q}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{f}*} \\ \mathbf{Q}_{*\mathbf{f}} & \mathbf{K}_{**} \end{bmatrix} \right), \tag{1.23}$$

which is the same as (1.22) except for the covariance of $\mathbf{f}_*$. This modification yields the following predictive equations:

$$\begin{aligned} p_{\text{PLV}}(y_*|x_*,\mathcal{D}) &= \mathcal{N}(y_*|\mu_{\text{PLV}*},\sigma^2_{\text{PLV}*}) \\ \mu_{\text{PLV}*} &= \mathbf{k}_{\mathbf{u}*}^\top(\mathbf{K}_{\mathbf{fu}}^\top\mathbf{K}_{\mathbf{fu}}+\sigma^2\mathbf{K}_{\mathbf{uu}})^{-1}\mathbf{K}_{\mathbf{fu}}^\top\mathbf{y} \\ \sigma^2_{\text{PLV}*} &= \sigma^2+\sigma^2\mathbf{k}_{\mathbf{u}*}^\top(\mathbf{K}_{\mathbf{fu}}^\top\mathbf{K}_{\mathbf{fu}}+\sigma^2\mathbf{K}_{\mathbf{uu}})^{-1}\mathbf{k}_{\mathbf{u}*}^\top \\ &\quad + k_{**}-\mathbf{k}_{\mathbf{u}*}^\top\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}*}\,, \end{aligned}$$

which keep the predictive mean of SR while healing its predictive variance. The two last extra terms of the predictive variance of y_* represent the predictive variance of $\mathbf{x}$ given $\mathbf{u}$, which makes up for the loss of prior variance in $k_{\text{SR}}(\mathbf{x},\mathbf{x})$. Since the effective covariance function used for training and test data is different, the consistency property of GP models is lost and PLV do not correspond exactly to a GP. They do, however, correspond to a proper probabilistic model in which $\mathbf{f}$ and $\mathbf{f}_*$ are treated differently, see Quiñonero-Candela and Rasmussen (2005). Computational complexity and storage needs are the same as for SR.

The downside of this approximation is that the selection of the active set and the hyperparameters interfere with each other, as noted in Seeger et al. (2003). This happens because the optimization of the marginal likelihood wrt the hyperparameters must be interleaved with the selection of the active set, and reselecting the active set causes non-smooth fluctuations in the marginal likelihood, which in turn jeopardize convergence.

1.2.5 Sparse pseudo-Input Gaussian Processes

The above proposals have been superseded by the Sparse Pseudo-inputs GP (SPGP) model, introduced in Snelson and Ghahramani (2006). SPGP represents the state of the art in approximate GPs, so it will be used as a benchmark for the approximations developed in this thesis.

One of the novelties of this model is that the constraint that the samples of the active set (which are called *pseudo-inputs* in this context) must be selected among training data is relaxed, allowing them to lie anywhere in the input space. This permits both pseudo-inputs and hyperparameters to be selected in a joint continuous optimization and increases flexibility, resulting in much superior performance.

SPGP was later renamed to Fully Independent Training Conditional (FITC) model to fit in the systematic framework of Quiñonero-Candela and Rasmussen (2005). Since the sparse model introduced in Chapter 4 also uses a fully independent training conditional, we will stick to the first name when referring to this method to avoid possible confusion.

The key idea of SPGP is to augment the existing training data set $\mathcal{D}$ with a noiseless pseudo-data set $\overline{\mathcal{D}} \equiv \{\overline{\mathbf{X}}, \mathbf{u}\}$ with pseudo-inputs $\overline{\mathbf{X}} = [\overline{\mathbf{x}}_1 \dots \overline{\mathbf{x}}_m]$ and pseudo-outputs $\mathbf{u} = [u_1 \dots u_m]^\top$ (which is smaller than the original data set, $m \ll n$) and assume that all latent variables (either from the train or test set) are conditionally independent given the pseudo-data set. The pseudo-data set consists therefore of m additional locations $\overline{\mathbf{X}}$ and their corresponding latent function values (the inducing variables $\mathbf{u}$). The pseudo-outputs can be integrated out and the pseudo-inputs learned maximizing the evidence of the model.

Conditioning on the pseudo data set, the marginal conditional distribution of a single latent function value f_j can be obtained:

$$p(f_j|\mathbf{x}_j, \overline{\mathcal{D}}) = \mathcal{N}(f_j|\mathbf{k}_j\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, k_j - q_j)$$

where $\mathbf{k}_j$ is the j-th row of $\mathbf{K}_{\mathbf{fu}}$, k_j is the j-th element of the diagonal of $\mathbf{K}_{\mathbf{ff}}$ and q_j is the j-th element of the diagonal of $\mathbf{Q}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}$. The expression for the conditional distribution of a test latent value is analogous.

Assuming conditional independence, it is possible to replace the joint conditional distribution of the latent values with the product of the marginal conditional distributions and thus the SPGP approximation arises:

$$p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \overline{\mathbf{X}}, \mathbf{u}) \approx \prod_{j=1}^{n} p(f_j|\mathbf{x}_j, \overline{\mathbf{X}}, \mathbf{u}) \prod_{j=1}^{n} p(f_{*j}|\mathbf{x}_{*j}, \overline{\mathbf{X}}, \mathbf{u}).$$

The original prior over the latent variables (1.11) is thus approximated by:

$$\begin{aligned}
p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \overline{\mathbf{X}}) &= \int p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \overline{\mathbf{X}}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u} \\
&\approx \int \prod_{j=1}^{n} p(f_j|\mathbf{x}_j, \overline{\mathbf{X}}, \mathbf{u}) \prod_{j=1}^{n} p(f_{*j}|\mathbf{x}_{*j}, \overline{\mathbf{X}}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u} \\
&= p_{\text{SPGP}}(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \overline{\mathbf{X}}) \\
&= \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{Q}_{\mathbf{ff}} + \operatorname{diag}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}) & \mathbf{Q}_{\mathbf{f}*} \\ \mathbf{Q}_{*\mathbf{f}} & \mathbf{Q}_{**} + \operatorname{diag}(\mathbf{K}_{**} - \mathbf{Q}_{**}) \end{bmatrix} \right).
\end{aligned}$$

where $\mathrm{diag}(\cdot)$ is an operator that sets all off-diagonal elements to zero.

In the typical setting where a single test case is considered at a time, the only difference between this approximate prior and that of PLV (1.23) is the extra diagonal matrix $\mathrm{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})$ in the training data covariance matrix, which has the effect of replacing the approximate diagonal elements with the original, exact ones. According to Snelson and Ghahramani (2006), this difference makes it much easier to select the pseudo-inputs by maximizing the evidence using conjugate gradients. Adding a diagonal matrix to the covariance matrix can be interpreted as the adding heteroscedastic white noise to the GP, enabling SPGP to model data sets with input-dependent levels of noise.

The pseudo-inputs and the hyperparameters of the model can be jointly selected by maximizing the evidence. For the ARD SE covariance function, this results in an optimization over $(m+1)D + 2$ real values. Optimization is usually performed using conjugate gradient methods, since gradients can be computed analytically.

This approximation is equivalent to a standard GP with covariance function:

$$k_{\text{SPGP}}(\mathbf{x}, \mathbf{x}') = \mathbf{k_u}(\mathbf{x})^\top \mathbf{K_{uu}^{-1}} \mathbf{k_u}(\mathbf{x}')(1 - \delta_{\mathbf{x}\mathbf{x}'}) + k(\mathbf{x}, \mathbf{x}')\delta_{\mathbf{x}\mathbf{x}'},$$

where Kronecker delta $\delta_{\mathbf{x}\mathbf{x}'}$ equals one if $\mathbf{x} = \mathbf{x}'$ and zero otherwise. The predictive equations are:

$$\begin{aligned} p_{\text{SPGP}}(y_*|x_*, \mathcal{D}) &= \mathcal{N}(y_*|\mu_{\text{SPGP}*}, \sigma^2_{\text{SPGP}*}) \\ \mu_{\text{SPGP}*} &= \mathbf{k_{u*}^\top}(\mathbf{K_{uu}} + \mathbf{K_{fu}^\top})^{-1}\mathbf{K_{fu}^\top}\mathbf{\Lambda_y^{-1}}\mathbf{y} \\ \sigma^2_{\text{SPGP}*} &= \sigma^2 + k_{**} + \mathbf{k_{u*}^\top}((\mathbf{K_{uu}} + \mathbf{K_{fu}^\top})^{-1} - \mathbf{K_{uu}^{-1}})\mathbf{k_{u*}}\,, \end{aligned}$$

where $\mathbf{\Lambda_y} = \mathrm{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2\mathbf{I}_n$. The computational complexity of SPGP is the same as that of SR and PLV.

1.2.6 Other approximations

Other probabilistically-founded approximations not covered here include Csató and Opper (2002), which develop an online version of PLV and the Bayesian Committee Machine (BCM) of Tresp (2000), in which predictions from different clusters of training data are combined. The BCM needs the location of the test samples $\mathbf{X}_*$ to be available at training time, making this approach transductive rather than inductive.

A different class of computationally efficient algorithms arises from the use of numerical approximations. It was suggested by Gibbs (1997) that the product of the inverse of the covariance matrix and an arbitrary vector could be computed as the solution of a linear system, using a conjugate gradient (CG) solver. This is a convex problem known to converge in n steps. If only m steps of the CG solver are actually run, an approximate solution can be obtained in $\mathcal{O}(mn^2)$ time. The prediction equations for GPs can then be expressed in terms of this basic matrix-vector product, rendering the overall computational cost $\mathcal{O}(mn^2)$. Note that this is in contrast with previous methods, which only needed $\mathcal{O}(m^2n)$ computation time. Other approximations involving some fast way to perform matrix-vector multiplications include Shen et al. (2006) and Yang et al. (2005).

1.3 Overview of the rest of the thesis

The objective of this thesis is to propose and test some new ideas to build sparse GP models that achieve, to some level, the benefits of full GPs and, at the same time, are able to handle large-scale data sets.

We will restrict ourselves to the development and analysis of models that can be trained (including model selection) in $\mathcal{O}(m^2n)$ time, can make probabilistic predictions in $\mathcal{O}(m^2)$ time per test case, and require $\mathcal{O}(mn)$ storage space, i.e., scale linearly with the number of training data[5]. The dependence of computation time on the dimensionality of data is linear and will be omitted in general. These complexity orders will make state-of-the-art SPGP an adequate benchmark, since it incurs in the same costs. Additional care will be taken for each method to also match the constant multiplicative factor of SPGP, so that computation time for all methods is roughly the same.

All the sparse GP models proposed in this thesis are proper GPs and fit in the framework of Quiñonero-Candela and Rasmussen (2005). Therefore, they can be interpreted either as approximations to full GPs or as GPs in their own right with a modified, computationally advantageous prior. In our exposition we will favor the latter, more recent view.

In Chapter 2 we introduce the Sparse Spectrum GP (SSGP), a stationary trigonometric Bayesian model that can be used to approximate any stationary full GP. Unlike

[5]Recall that for full GPs computation time scales cubically, and storage space, quadratically.

previous sparse models, SSGP is a truly stationary process (not only an *approximation* of a stationary process). SSGP is first presented as a Monte Carlo approximation to a full GP and then two alternative Bayesian interpretations are provided. If the constraint of convergence to the full GP in the infinite limit is dropped, the flexibility of SSGP can be increased, yielding very competitive results on large-scale regression problems.

In Chapter 3 we generalize SSGP to a broader class of Bayesian models with the structure of generalized linear models where the "output weights" have been marginalized out. We call them Marginalized Networks (MNs). We investigate the drawbacks of the direct application of MNs to regression problems and we propose two different ways to overcome them: Noise bounding and network mixing. We check the usefulness of these improved MNs on large-scale problems and compare them with SSGP. Finally, we show how to take advantage of the structure of these networks to perform linear dimensionality reduction.

In Chapter 4 we extend the index set of GPs to include other domains. We call the resulting processes Inter-Domain GPs (IDGPs). By defining a GP over more than one domain, it is possible to extend SPGP and to place the "pseudo-inputs" in other domains[6]. This has two interesting effects: It detaches the form of the basis functions from the form of the covariance function, and it adds extra flexibility that results in improved performance. This is a general framework which includes other previously developed sparse models such as SPGP or the Sparse Multi-Scale GPs from Walder et al. (2008), giving further insights on how they work. IDGPs can also be used for other purposes not related to efficient inference, such as inference across domains or constraining the latent function, but we do not pursue these possibilities here.

In Chapter 5 we investigate possible extensions of the previous ideas. First, we cast Multi-Layer Perceptrons as MNs and test their performance. Then we provide details on how sparse GP models with certain covariance structure (including, but not restricted to, those described in this thesis) can be efficiently extended to handle non-Gaussian likelihoods. Finally, extensions for robust regression and classification are developed and tested on the corresponding data sets.

Chapter 6 concludes this work with a summary of the contributions, a comparison chart of the proposed models and a brief discussion about further work.

[6]Since pseudo-inputs will no longer be in the input domain, we refer to them as "inducing features" in this context.

Chapter 2

Sparse Spectrum GPs

In this chapter we introduce a stationary trigonometric Bayesian regression model which retains the computational efficiency of aforementioned approaches such as SR, PLV or SPGP (Sections 1.2.3-1.2.5), while improving the accuracy of the predictions. The model consists of a weighted sum of trigonometric functions where both weights and phases are integrated out. Frequencies can be both fixed or learned. Hyperparameters (and possibly frequencies) are learned by jointly maximizing the evidence of the model. We will show that this model is a stationary sparse GP that approximates any stationary full GP, provided that the frequencies follow a given distribution. This approximation improves with the number of frequencies and converges to the full GP in the infinite limit. A technical report with a preliminary version of our results was published in Lazaro-Gredilla et al. (2007). Shortly after this report, Rahimi and Recht (2008) proposed a related approximation in the context of Support Vector Machines using a finite set of random Fourier features. In that work the frequencies were selected randomly, in contrast to our work where these quantities can also be learned.

This chapter is organized as follows: We will derive the model from different equivalent perspectives in Section 2.1. The properties of the model are discussed in Section 2.2. Model selection is described in Section 2.3. In Section 2.4, the performance of the model is investigated on several data sets and compared to SPGP. Some thoughts regarding overfitting are discussed in Section 2.5. The implications of learning the phases versus integrating them out are considered in Section 2.6. We wrap-up in Section 2.7.

2.1 The model: Sparse Spectrum GP (SSGP)

The Sparse Spectrum GP is based on a sparse approximation to the frequency domain representation of a GP. It can be derived and presented from different equivalent perspectives. In this thesis we have chosen to start off introducing it as a numerical Monte Carlo approximation to a full GP and then provide the equivalent Bayesian perspective as a trigonometric model.

2.1.1 SSGP as a Monte Carlo approximation to a full GP

The power spectral density (or power spectrum) $S(\mathbf{s})$, with $\mathbf{s} \in \mathbb{R}^D$, of a stationary random function tells us how the power is distributed over the frequency domain. The total power over the whole frequency domain is equal to the prior pointwise variance $k(\mathbf{0}) = \sigma_0^2$. Both frequency and input domains have the same dimension, D, and the d-th element of $\mathbf{s}$ can be interpreted as the frequency associated to the d-th input dimension. The Wiener-Kintchine theorem (see for example Carlson (1986, p. 162)) tells us that the power spectrum and the autocorrelation of a random function constitute a Fourier pair. In our case, given that $f(\cdot)$ is drawn from a stationary Gaussian process, the autocorrelation function is equal to the stationary covariance function, and we have:

$$k(\boldsymbol{\tau}) = \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top \boldsymbol{\tau}} S(\mathbf{s}) d\mathbf{s}, \tag{2.1}$$

$$S(\mathbf{s}) = \int_{\mathbb{R}^D} e^{-2\pi i \mathbf{s}^\top \boldsymbol{\tau}} k(\boldsymbol{\tau}) d\boldsymbol{\tau}. \tag{2.2}$$

We thus see that there are two equivalent representations for a stationary GP: The traditional one in terms of the covariance function in the input domain, and a perhaps less usual one as the power spectrum in the frequency domain.

Bochner's theorem —Stein (1999, p. 24)— states that any stationary covariance function $k(\boldsymbol{\tau})$ can be represented as the Fourier transform of a positive finite measure. This means that the power spectrum (2.2) is a positive finite measure, and in particular that it is *proportional* to a probability measure, $S(\mathbf{s}) \propto p_S(\mathbf{s})$. The proportionality constant can be directly obtained by evaluating (2.1) at $\boldsymbol{\tau} = \mathbf{0}$. We obtain the relation:

$$S(\mathbf{s}) = k(\mathbf{0})\, p_S(\mathbf{s}) = \sigma_0^2\, p_S(\mathbf{s}). \tag{2.3}$$

We can use the fact that $S(\mathbf{s})$ is proportional to a multivariate probability density in $\mathbf{s}$ to rewrite the covariance function in (2.1) as an expectation:

$$\begin{aligned} k(\mathbf{x},\mathbf{x}') = k(\boldsymbol{\tau}) &= \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top(\mathbf{x}-\mathbf{x}')} S(\mathbf{s})\,\mathrm{d}\mathbf{s} = \sigma_0^2 \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top \mathbf{x}} \left(e^{2\pi i \mathbf{s}^\top \mathbf{x}'}\right)^* p_S(\mathbf{s}) d\mathbf{s} \\ &= \sigma_0^2 \mathbb{E}_{p_S}\left[e^{2\pi i \mathbf{s}^\top \mathbf{x}} \left(e^{2\pi i \mathbf{s}^\top \mathbf{x}'}\right)^*\right], \end{aligned} \tag{2.4}$$

where $\mathbb{E}_{p_S}$ denotes expectation wrt $p_S(\mathbf{s})$ and superscript asterisk denotes complex conjugation (not to be confused with the subscript asterisk indicating test quantity). This last expression is an exact expansion of the covariance function as the expectation of a product of complex exponentials with respect to a particular distribution over their frequencies. Instead of exactly evaluating this average by integrating over all values of $\mathbf{s}$, we propose to obtain a Monte Carlo estimate by taking the average of a few samples corresponding to a finite set of frequencies, which we call *spectral points*.

Since we are dealing with real-valued functions, covariance function $k(\boldsymbol{\tau})$ can only take real values as well. By the properties of the Fourier transform, this implies that $S(\mathbf{s}) = S(-\mathbf{s})$ and therefore, $p_S(\mathbf{s}) = p_S(-\mathbf{s})$. It is then possible to draw samples from $p_S(\mathbf{s})$ in pairs of the form $\{\mathbf{s}_r, -\mathbf{s}_r\}$ and still have a valid Monte Carlo procedure. The advantage of drawing samples in symmetric pairs is that we preserve the property of exact expansion (2.4) that the imaginary terms cancel out. Drawing h sample pairs and evaluating the product of complex exponentials at them, we can approximately reconstruct the covariance function as a finite sum:

$$\begin{aligned} k(\mathbf{x},\mathbf{x}') \simeq k_{\text{SSGP}}(\mathbf{x},\mathbf{x}') &= \frac{\sigma_0^2}{2h}\sum_{r=1}^{h}\left[e^{2\pi i \mathbf{s}_r^\top \mathbf{x}}\left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}'}\right)^* + e^{2\pi i(-\mathbf{s}_r)^\top \mathbf{x}}\left(e^{2\pi i(-\mathbf{s}_r)^\top \mathbf{x}'}\right)^*\right] \\ &= \frac{\sigma_0^2}{2h}\sum_{r=1}^{h}\left[e^{2\pi i \mathbf{s}_r^\top \mathbf{x}}\left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}'}\right)^* + \left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}}\right)^* e^{2\pi i \mathbf{s}_r^\top \mathbf{x}'}\right] \\ &= \frac{\sigma_0^2}{h}\mathrm{Re}\left[\sum_{r=1}^{h} e^{2\pi i \mathbf{s}_r^\top \mathbf{x}}\left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}'}\right)^*\right], \end{aligned} \tag{2.5}$$

where $\mathrm{Re}\,[\cdot]$ denotes the real part of a complex number. The approximation becomes exact when h tends to infinity, since we recover (2.4). Note that only one spectral point out of each symmetric pair appears in the final expression.

In order to avoid working with complex numbers we expand the complex exponential in sines and cosines. Defining a column vector of length $2h$ containing the evaluation of the h pairs of trigonometric functions at $\mathbf{x}$

$$\boldsymbol{\phi}(\mathbf{x}) = \left[\cos(2\pi \mathbf{s}_1^\top \mathbf{x})\ \sin(2\pi \mathbf{s}_1^\top \mathbf{x})\ \ldots\ \cos(2\pi \mathbf{s}_m^\top \mathbf{x})\quad \sin(2\pi \mathbf{s}_m^\top \mathbf{x}_1)\right]^\top, \tag{2.6}$$

we can express the approximate covariance function (2.5) as a dot product:

$$k_{\text{SSGP}}(\mathbf{x}, \mathbf{x}') \;=\; \frac{\sigma_0^2}{h}\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}') \;=\; \frac{\sigma_0^2}{h}\sum_{r=1}^{h} \cos\left(2\pi \mathbf{s}_r^\top (\mathbf{x}-\mathbf{x}')\right), \tag{2.7}$$

which is a stationary covariance function, with constant pointwise variance σ_0^2.

This approximation is equivalent to replacing the original spectrum $S(\mathbf{s})$ by a set of h pairs of Dirac deltas of amplitude $\frac{\sigma_0^2}{2h}$ distributed according to $S(\mathbf{s})/\sigma_0^2$. This is why this approach is said to "sparsify" the spectrum of the GP.

This yields the following low rank approximation to the covariance matrix:

$$\mathbf{K}_{\mathbf{ff}} \;\simeq\; \frac{\sigma_0^2}{h}\boldsymbol{\Phi}_{\mathbf{f}}^\top \boldsymbol{\Phi}_{\mathbf{f}}, \tag{2.8}$$

where $\boldsymbol{\Phi}_{\mathbf{f}} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)]$ is a $2h \times n$ matrix, usually referred to as the "design matrix". Substituting this approximation in (1.17), the marginal likelihood of the model is obtained

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \frac{\sigma_0^2}{h}\boldsymbol{\Phi}_{\mathbf{f}}^\top \boldsymbol{\Phi}_{\mathbf{f}} + \sigma^2\mathbf{I}\right). \tag{2.9}$$

The predictions and marginal log-likelihood can be evaluated using eqs. (1.15) and (1.18), although direct evaluation is computationally inefficient when $2h < n$. Using matrix inversion lemma (A.1), we can express the predictive distribution efficiently as:

$$p_{\text{SSGP}}(y_*|x_*, \mathcal{D}) = \mathcal{N}(y_*|\mu_{\text{SSGP}*}, \sigma^2_{\text{SSGP}*}) \tag{2.10a}$$

$$\mu_{\text{SSGP}*} = \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}^{-1}\boldsymbol{\Phi}_{\mathbf{f}}\mathbf{y} \tag{2.10b}$$

$$\sigma^2_{\text{SSGP}*} = \sigma^2 + \sigma^2\boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_*), \tag{2.10c}$$

where we have defined $\mathbf{A} = \boldsymbol{\Phi}_{\mathbf{f}}\boldsymbol{\Phi}_{\mathbf{f}}^\top + \frac{h\sigma^2}{\sigma_0^2}\mathbf{I}_{2h}$. This expression clarifies that the posterior mean is a linear combination of $m = 2h$ basis functions. Similarly, using (A.1) and (A.2), an efficient expression to compute the negative log marginal likelihood is obtained:

$$\begin{aligned} -\log p_{\text{SSGP}}(\mathbf{y}|\mathbf{X}, \theta) = &\left[\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\boldsymbol{\Phi}_{\mathbf{f}}^\top \mathbf{A}^{-1}\boldsymbol{\Phi}_{\mathbf{f}}\mathbf{y}\right]/(2\sigma^2) \\ &+ \frac{1}{2}\log|\mathbf{A}| - h\log\frac{h\sigma^2}{\sigma_0^2} + \frac{n}{2}\log 2\pi\sigma^2. \end{aligned} \tag{2.11}$$

The actual implementation of these equation should be done via Cholesky decomposition, as detailed in Section D.2 of Appendix D. Both the predictive distribution

and the marginal likelihood can be computed in $\mathcal{O}(nm^2)$ (recall that $m = 2h$ is the number of basis functions). Predictive mean and variance at additional test data points can be computed in $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ respectively. The storage costs are also reduced, since we no longer store the full covariance matrix (of size $n \times n$), but only the design matrix (of size $n \times 2h$). Storage needs are then $\mathcal{O}(nm)$.

2.1.2 SSGP as a trigonometric Bayesian model

In this section we will describe Bayesian inference in a linear regression model with trigonometric basis functions and relate it to the GP approximation derived in the previous section.

2.1.2.1 The sine-cosine model

Consider the following model for the latent function

$$f(\mathbf{x}) = \sum_{r=1}^{h} \left[a_r \cos(2\pi \mathbf{s}_r^\top \mathbf{x}) + b_r \sin(2\pi \mathbf{s}_r^\top \mathbf{x}) \right] , \tag{2.12}$$

where the basis functions are parametrized by the m D-dimensional vectors $\{\mathbf{s}_r\}_{r=1}^h$, containing spectral frequencies. Note that each *pair* of basis functions shares frequencies, but each has independent amplitude parameters, a_r and b_r. We will treat the frequencies as deterministic parameters and the amplitudes in a Bayesian way. The priors on the weights are independent Gaussian

$$p(a_r) = \mathcal{N}(a_r|0, \sigma_0^2/h) , \quad p(b_r) = \mathcal{N}(b_r|0, \sigma_0^2/h) ,$$

where the variances are scaled down linearly by the number of basis functions.

By re-arranging the weights as a single vector $\mathbf{w} = [a_1,\ b_1,\ a_2,\ b_2,\ \ldots,\ a_h,\ b_h]^\top$ and evaluating the latent function at the training points $\mathbf{X}$, we can rewrite the linear model in vectorial form, $\mathbf{f} = \mathbf{\Phi_f w}$, and the prior is then

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \sigma_0^2 \mathbf{I}_{2h}/h).$$

The observed outputs are obtained from the latent values by adding noise $\mathbf{y} = \mathbf{f} + \mathbf{n}$ with $p(\mathbf{n}) = \mathcal{N}(\mathbf{n}|\mathbf{0}, \sigma^2 \mathbf{I}_n)$, so that the likelihood is

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}\left(\mathbf{y}|\mathbf{\Phi_f w}, \sigma^2 \mathbf{I}_n\right) . \tag{2.13}$$

The marginal likelihood of the model is then

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})d\mathbf{w} = \int \mathcal{N}\left(\mathbf{y}|\mathbf{\Phi_f}\mathbf{w},\sigma^2\mathbf{I}_n\right)\mathcal{N}(\mathbf{w}|\mathbf{0},\sigma_0^2\mathbf{I}_{2h}/h)d\mathbf{w} \\ &= \mathcal{N}(\mathbf{y}|0,\sigma_0^2\mathbf{\Phi_f}^\top\mathbf{\Phi_f}/h+\sigma^2\mathbf{I}_n), \end{aligned} \tag{2.14}$$

where (B.4) can be used to solve the integral.

The posterior over the weights can be obtained using Bayes rule

$$p(\mathbf{w}|\mathcal{D}) = p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} = \mathcal{N}(\mathbf{w}|\mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y},\sigma_n^2\mathbf{A}^{-1}) \ , \tag{2.15}$$

and the predictive posterior at some test point $\mathbf{x}_*$, using (2.13) and (2.15), is

$$\begin{aligned} p_{\text{SSGP}}(y_*|x_*,\mathcal{D}) &= \int p(y_*|\mathbf{x}_*,\mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \\ &= \int \mathcal{N}(y_*|\boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{w},\sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y},\sigma_n^2\mathbf{A}^{-1})d\mathbf{w} \\ &= \mathcal{N}(y_*|\boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y},\ \sigma^2+\sigma^2\boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_*)), \end{aligned} \tag{2.16}$$

where the integral is again easily solved using (B.4).

Eqs. (2.9) and (2.14) describing $p(\mathbf{y}|\mathbf{X})$ are identical, thus proving that the Monte Carlo approximation and the Bayesian combination of trigonometric functions are equivalent models. The predictive distributions obtained in both cases are necessarily identical too, as (2.10) and (2.16) demonstrate.

We have thus provided two alternative derivations of the SSGP model. The Bayesian derivation additionally provides posterior densities over the amplitudes of each frequency, whereas the connection with full GPs and the infinite limit convergence is more clearly shown in the Monte Carlo derivation.

This derivation shows that a combination of sines and cosines with random, Gaussian distributed amplitudes is equivalent to a full GP *with arbitrary stationary covariance* as the number of basis tends to infinity, if their frequencies (i.e. the spectral points $\{\mathbf{s}_r\}_{r=1}^h$) are distributed according to the spectral density of that stationary covariance function. A detailed proof is given in Section C.2 of Appendix C. This is a more general result than that of MacKay (2003, ch. 45), which only applies to the ARD SE covariance function.

2.1.2.2 The cosine-phase model

There is yet another way to express the SSGP model within the Bayesian framework. Consider the following regression model:

$$f(\mathbf{x}_j) \;=\; \sum_{r=1}^{h} c_r \cos(2\pi \mathbf{s}_r^\top \mathbf{x}_j - \varphi_r) \;. \tag{2.17}$$

where $\{c_r\}_{r=1}^h$ and $\{\varphi_r\}_{r=1}^h$ are amplitude and phase parameters. As before, we consider the spectral points $\{\mathbf{s}_r\}_{r=1}^h$ as given deterministic parameters.

Following a proper Bayesian treatment, we place a prior on the parameters of the model, $\{c_r\}_{r=1}^h$ and $\{\varphi_r\}_{r=1}^h$. For the phases, the most reasonable prior is a uniform distribution, since no concrete value is to be preferred. Since the cosine is 2π periodic, any uniform distribution covering a 2π range accounts for all possible phase values. We will therefore use

$$p(\varphi_r) \;=\; \begin{cases} \frac{1}{2\pi} & \text{if} \quad -\pi < \varphi_r < \pi \\ 0 & \text{otherwise} \;. \end{cases} \tag{2.18}$$

For the weights we will use a symmetric-Rayleigh distribution:

$$p(c_r) \;=\; \frac{1}{2v}|c_r| \exp\left(-\frac{c_r^2}{2v}\right) \;, \quad \text{with} \quad v \;=\; \frac{\sigma_0^2}{h} \;. \tag{2.19}$$

In order to be able to conveniently integrate out both weights and phases from the model without resorting to any approximation, we make the following variable change:

$$a_r \;=\; c_r \cos\varphi_r \;\text{ and }\; b_r \;=\; c_r \sin\varphi_r \;. \tag{2.20}$$

that allows us to rewrite our model as follows:

$$f(\mathbf{x}) = \sum_{r=1}^{h} \left[a_r \cos(2\pi \mathbf{s}_r^\top \mathbf{x}) + b_r \sin(2\pi \mathbf{s}_r^\top \mathbf{x})\right] \;.$$

Given the independent priors placed on $\{c_r\}_{r=1}^h$ and $\{\varphi_r\}_{r=1}^h$ and the relation established in (2.20), we prove in Section C.1 of Appendix C that $\{a_r\}_{r=1}^h$ and $\{b_r\}_{r=1}^h$ are also independent and Gaussian distributed. In this case their distributions are

$$p(a_r) = \mathcal{N}(a_r|0, \sigma_0^2/h) \;, \quad p(b_r) = \mathcal{N}(b_r|0, \sigma_0^2/h) \;,$$

so the sine-cosine model (2.12), with the same priors, is retrieved.

The importance of this derivation is to show that SSGP corresponds to a trigonometric model in which phases have been integrated out. Unlike most sparse GP models, no location hyperparameters need to be selected in SSGP. This results in enhanced robustness. We will investigate in more detail the effect of fixing/integrating out the phases in Section 2.6.

It is also interesting to note that predictive mean (2.10b) can be expressed either as a linear combination of sines and cosines (without phase) or as a linear combination of cosines (with phase), in line with both equivalent models (2.12) and (2.17). We can illustrate this by first computing coefficients $[a'_1,\ b'_1,\ \ldots,\ a'_h,\ b'_h]^\top = \mathbf{A}^{-1}\mathbf{\Phi}_\mathbf{f}\mathbf{y}$ and then re-expressing the predictive mean in both ways

$$\begin{aligned}\mu_{\text{SSGP}*} &= \sum_{r=1}^{h}\left[a'_r\cos(2\pi\mathbf{s}_r^\top\mathbf{x}_*) + b'_r\sin(2\pi\mathbf{s}_r^\top\mathbf{x}_*)\right] \\ &= \sum_{r=1}^{h}\sqrt{a_r'^2+b_r'^2}\cos(2\pi\mathbf{s}_r^\top\mathbf{x}_* - \arctan(b'_r/a'_r))\ ,\end{aligned}$$

where $\arctan\frac{b'_r}{a'_r}$ is the angle corresponding to coordinates (a'_r, b'_r).

2.1.3 Example: the ARD SE covariance case

The probability density associated to ARD SE covariance function (1.5) can be expressed in the form

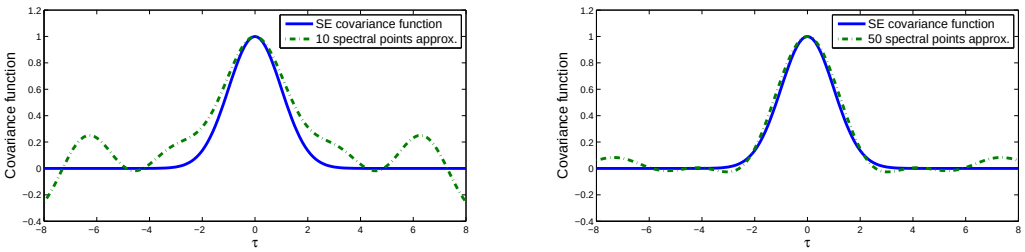
$$p_S^{\text{ARDSE}}(\mathbf{s}) = \frac{1}{k_{\text{ARDSE}}(\mathbf{0})}\int_{\mathbb{R}^D} e^{-2\pi i\mathbf{s}^\top\boldsymbol{\tau}}k_{\text{ARDSE}}(\boldsymbol{\tau})d\boldsymbol{\tau} = \sqrt{|2\pi\mathbf{L}^2|}\exp\left(-\frac{1}{2}(\mathbf{s}^\top(2\pi\mathbf{L})^2\mathbf{s}\right)$$

where $\mathbf{L}$ is a diagonal matrix of size $D\times D$ whose elements are the length-scales $\ell_1,\ \ldots\ \ell_D$.

Spectral points following this distribution can be generated in two steps: First, h random vectors $\{\boldsymbol{\omega}_r\}_{r=1}^h$ are drawn from a normal distribution $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0},\mathbf{I}_D)$. Then, they are scaled to produce the spectral points $\mathbf{s}_r = (2\pi\mathbf{L})^{-1}\boldsymbol{\omega}_r$. The advantage of this two-step procedure is that we can learn only the length-scales $\{\ell_d\}_{d=1}^D$ (keeping $\{\boldsymbol{\omega}_r\}_{r=1}^h$ fixed) or every single spectral point (learning both $\{\ell_d\}_{d=1}^D$ and $\boldsymbol{\omega}_r$).

For illustration purposes, we compare the exact squared exponential covariance function with its spectral approximation in Figure 2.1. The spectral points have been generated randomly from the appropriate probability density, which in this case is also

Gaussian. The approximation is more accurate around the origin, in a range that can be extended by using more spectral points. Note that even if the number of spectral points is too low to achieve a good approximation within a wide range, the resulting covariance function can still be good enough for most purposes.



(a) 10 spectral points

(b) 50 spectral points

Figure 2.1: Squared exponential covariance function and its reconstruction from a few spectral samples.

For low dimensions, the quality of the reconstruction can be good enough even if a small number of points is used. For higher dimensions, the number of spectral points required to cover equally well the spectrum is much larger (since the volume of the space grows exponentially with the dimension). In this case, to be able to take the most representative samples of the spectrum with just a few points we need to resort to learning their locations.

If $\{\boldsymbol{\omega}_r\}_{r=1}^h$ are learned, the probability density that they represent may also change, which in turn changes the covariance function being approximated. This additional flexibility both improves performance (since we are effectively learning the covariance function) and increases the risk of overfitting. In practice, when working with large data sets and a number of basis functions $m \ll n$, overfitting rarely occurs. In Chapter 3 we present and explore two methods for enhanced overfitting resistance.

In Section 2.3 we will describe model selection, suggesting to initialize the spectral points to approximate the ARD SE covariance function, drawing them from $p_S^{\text{ARD SE}}(\mathbf{s})$ as derived here. Of course, they can be initialized to approximate any stationary covariance function, using the corresponding probability density.

2.2 SSGP properties

In this section, we will review some unique properties that set SSGP apart from other sparse GP models.

2.2.1 Stationary nature

The most popular covariance function in GP regression is the ARD SE covariance function (1.5), which is stationary, i.e., it is only a function of the relative position of the inputs. This is very reasonable for many data sets, since it means that the degree of coupling between any two points does not depend on their absolute location within input space, but only on where they are located with respect to each other. Interestingly, the covariance function of SSGP is $k_{\text{SSGP}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_0^2}{h} \sum_{r=1}^{h} \cos\left(2\pi \mathbf{s}_r^\top (\mathbf{x} - \mathbf{x}')\right)$ which depends only on the difference of the inputs, and is therefore stationary.

This property, which is unique to SSGP within existing sparse GP models, means that no matter how the spectral points $\{\mathbf{s}_r\}_{r=1}^h$ are arranged, the covariance between points with the same relative position will be the same. Conversely: Spectral points can only be arranged in a way that fits pairwise interactions all throughout the model, so as to maximize the evidence. This constraint helps to reduce overfitting in cases where many spectral points are selected.

We can look at the effect of stationarity from another perspective: From (1.14), we know that the predictive mean of a GP with some prior covariance $k(\cdot, \cdot)$, can be expressed as a summation of "kernels":

$$\mathbb{E}[f(\mathbf{x})] = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x});,$$

where n is the number of samples, $\{\mathbf{x}_j\}_{j=1}^n$ are the input data and $\{\alpha_j\}_{j=1}^n$ are scaling factors derived from the predictive equations for GP.

Each kernel $k(\mathbf{x}_j, \mathbf{x})$ is said to be "centered" at the input sample $\mathbf{x}_j$. The only difference between all kernels is a shifting, with shapes being conserved, *only if* $k(\cdot, \cdot)$ *is stationary*. If $k(\cdot, \cdot)$ is non-stationary, the shape of the kernels centered at different points may be different.

Since SSGP has a stationary covariance function, the interpolating function is composed of the summation of the *same shape* centered at the input points, with different

scales. That shape is constrained to provide a good fit to data when used all throughout the input space. Since the shape is solely determined by the arrangement of the spectral points, these are also constrained. When the locations of the spectral points are learned, stationarity implies some overfitting resistance: Every spectral point location is constrained by every input-target pair.

2.2.2 No location parameters

Since SSGP is only parametrized by the spectral points[1], no location parameters need to be adjusted. This is in contrast with most previous approximations, where some kind of input-space parametrization was always present: The pseudo-inputs in SPGP, active set selection in SR and PLV, etc. SSGP cannot get lost in irrelevant parts of the input space while trying to learn location parameters, which is sometimes a problem with SPGP.

2.2.3 Periodicity

One might be tempted to assume that this model would only be useful for modeling periodic functions (as the Bayesian derivation suggests), since strictly speaking a linear combination of periodic signals is itself periodic, but in practice periods can be made big enough to render it useful for the general regression case (which makes sense when considering the Monte Carlo derivation).

Periodicity implies that predictions made far away from the training data set can be different from zero, in contrast to models using localized basis functions. Furthermore, infinite replicas of the predictive mean and variance will appear in the input space, forming a grid. The spacing of this grid in each of the possible dimensions corresponds to the overall period of the model in that direction.

For any given dimension, the overall period of the model corresponds to the least common multiple of the individual periods of the constituting basis functions. Suppose for a moment that the frequencies (spectral points) of the basis functions were arranged in a grid, so that any spectral point could be expressed as $\mathbf{s}_r \equiv [t_{1r}s_1,\ t_{2r}s_2,\ \ldots, t_{Dr}s_D]$,

[1] It is also parametrized by $\{\ell_d\}_{d=1}^D$, σ_0^2 and σ^2 but this hyperparameters also appear in the full GP and are not related to input domain locations.

where $t_{dr} \in \mathbb{Z}$ and s_d is the frequency spacing for dimension d. In this case, the overall period for any single dimension would be $1/s_d$. Therefore, the finer the grid is made, the bigger the period becomes. If we want the model to cover some fixed range of frequencies, increasing the number of basis results in a finer grid, and replicas get spaced away.

Since we will not be arranging the spectral points in a grid, but randomly, previous analysis is pessimistic. The least common multiple of the periods corresponding to random frequencies is much bigger, but it is still true that grows with the density of the frequencies. The same principle is used (interchanging input and frequency domains) in uneven sampling to space apart frequency replicas an avoid aliasing, see for instance Bretthorst (2000).

In practice, replicas are sufficiently spaced even for a very moderate number of spectral points and thus the model has practical use for modeling non-periodic functions. However, we are warned that the model could produce meaningless predictive values if used for strong extrapolation. In Section 2.4.1 we will show an extrapolation example to understand the effects of periodicity.

2.2.4 Sparse Fourier Transform

A by-product of the inference process is a Sparse Fourier Transform of the multidimensional signal y, unevenly sampled at points $\{\mathbf{x}_j\}_{j=1}^n$. This Sparse Fourier Transform strongly resembles the model presented in Qi et al. (2002), with the important difference that whereas SSGP automatically selects a discrete set of frequencies, the model of the cited work assumes that these frequencies are given. Equation (2.15) provides the posterior distribution over the Fourier Transform at selected frequencies.

2.3 Model selection

In order to be able to effectively use this model, we need a method to select σ_0^2, σ_n^2 and $\{\ell_d\}_{d=1}^D$ (the hyperparameters of a standard GP) as well as the spectral points, represented in this case by $\boldsymbol{\omega}_r$, since $\mathbf{s}_r = (2\pi\mathbf{L})^{-1}\boldsymbol{\omega}_r$.

Following the discussion on Section 1.1.5, we will use Type II Maximum Likelihood to select these values. This implies selecting them so as to minimize the Negative Log Marginal Likelihood (NLML) of the model, which can be efficiently computed using (2.11).

2.3.1 SSGP with selectable spectral points

We will first consider the case in which the spectral points are learned along with the rest of the hyperparameters. It might seem at first that since $\mathbf{s}_r = (2\pi\mathbf{L})^{-1}\boldsymbol{\omega}_r$, we could avoid learning the length-scales $\{\ell_d\}_{d=1}^D$, considering them contained in $\boldsymbol{\omega}_r$ instead. However, when using practical optimization techniques, this over-parameterization proves useful, allowing to locate better minima of the NLML. In particular, this redundancy is critical for the model to correctly perform ARD. If a problem has some non-informative dimension, it can be pruned by setting the corresponding length-scale to a big value. Though this is equivalent to setting the corresponding component of *all* spectral points to a very small value, from an optimization point of view it is much easier to learn a single length-scale hyperparameter.

Model selection is therefore as follows:

1. Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, and σ^2 to some sensible values. (An example of this would be: one half of the ranges of the input dimensions, the variance of the outputs $\{y_j\}_{j=1}^n$ and $\sigma_0^2/4$, respectively).

2. Initialize $\{\boldsymbol{\omega}_r\}_{r=1}^h$ from $\mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ (to initially approximate the ARD SE covariance function).

3. Since analytical derivatives of the NLML wrt to all previous hyperparameters can be computed from (2.11), use conjugate gradient descent to jointly minimize NLML.

This method uses non-linear optimization to select the $D+2+hD$ hyperparameters of the model. In terms of basis functions this is $D+2+mD/2$, i.e., roughly one half the number of hyperparameters needed by SPGP, $D + 2 + mD$. The exact location of the spectral points (which account for the vast majority, hD, of the hyperparameters being selected) is not critical, as long as they cover the spectrum of the data set relatively

well. Equivalently, in SPGP, the exact location of the pseudoinputs is not determinant, as long as they are distributed covering the input space of the data set well enough. The remaining $D+2$ hyperparameters that account for the length-scales, signal power and noise power are more critical, but also much fewer, so that we can hope for them to be well determined from data. In fact, they are exactly the hyperparameters used by a standard GP with ARD SE covariance function. When using a very large number of spectral points some overfitting might appear, but this is not generally the case. In fact, SSGP seems to be quite resistant to overfitting as we will see in Section 2.4. Some theoretical justification for this was provided in Section 2.2.1.

It is possible to compute all of the $D+2+mD/2$ required NLML derivatives in $\mathcal{O}(m^2n)$ time. This is achieved using (A.6), (A.7) on (2.11) and then noticing that the m NLML derivatives wrt *all components corresponding to a single dimension of the spectral points* can be computed as a block in $\mathcal{O}(m^2n)$ time (see Section E.2 of Appendix E for details). Therefore, model selection does not cause any overhead to the complexity order mentioned above. Derivatives are computed at every iteration of the optimization process, so this cost often dominates the overall process when making predictions.

2.3.2 SSGP with fixed spectral points

Another possibility is to let $\{\boldsymbol{\omega}_r\}_{r=1}^h$ fixed at their initialization values, and learn only the remaining hyperparameters. This completely avoids any overfitting problems, but comes at the cost of needing a larger number of spectral points to reach the same performance. In the experimental section we will consider this case for comparison, and will refer to it as SSGP-fixed. For problems in one or two dimensions, this should be the method of choice, since the spectrum is already well covered with the initial values of $\{\boldsymbol{\omega}_r\}_{r=1}^h$ (remember that length-scales are still learned).

This option has the attractive property of converging to the full GP as the number of spectral samples grows, see Section C.2 of Appendix C. Observe that if the spectral points were learned, their overall distribution could be distorted, and convergence to the desired full GP would not be achieved. Additionally, this option is as immune to overfitting as a full GP, since both have the same number of hyperparameters, which is very small.

It should be emphasized that SPGP in its typical setting (i.e., with its pseudo-inputs being selected through ML-II) does *not* converge to the full GP as the number of basis function grows. For SPGP to converge to the full GP, the pseudo-inputs must be initialized to a subset of training data and kept fixed (which in turn degrades the performance of SPGP in the sparser regime).

2.4 Experiments

In this section we are going to thoroughly explore the properties and performance of SSGP[2], first on a toy problem and then on bigger, real world problems. We will use the current state of the art for sparse GP regression, SPGP[3], as benchmark for comparison and will quote the results obtained by a full GP as a reference. Both SPGP and the full GP use the ARD SE covariance function, and SSGP is initialized to approximate it.

Both approximations run in $\mathcal{O}(m^2 n)$. To match the constant multiplicative factor, we will use the same number of basis functions for both methods (this matches the size of the involved matrices, so that computational cost becomes roughly identical). In other words, SPGP will be using two pseudo-inputs per spectral point of SSGP. Actually, we have compared the speeds of SPGP and SSGP (both running on the same platform), and even using only one pseudo-input per spectral point, SSGP is still faster. This is due to SSGP being a simpler model, so that NLML derivatives and predictions can be expressed more compactly and are much faster to compute. Nonetheless, we will maintain the more conservative option of using the number of basis functions as the basis for comparison.

We will report the test Normalized Mean Square Error (NMSE) and the test Mean Negative Log Probability (MNLP) as performance measures. They are defined as:

$$\text{NMSE} = \frac{\sum_{j=1}^{n_*}(y_{*j} - \mu_{*j})^2}{\sum_{j=1}^{n_*}(y_{*j} - \overline{y})^2} \tag{2.21a}$$

$$\text{MNLP} = \frac{1}{2n_*}\sum_{j=1}^{n_*}\left[\left(\frac{y_{*j} - \mu_{*j}}{\sigma_{*j}}\right)^2 + \log\sigma_{*j}^2 + \log 2\pi\right] , \tag{2.21b}$$

where μ_{*j} and σ_{*j}^2 are, respectively, the predictive mean and variance for the j-th test output and y_{*j} is the actual test output value. The average of the training output values

[2]For our code implementing SSGP, check Appendix F.

[3]The implementation from its authors is used, see http://www.gatsby.ucl.ac.uk/~snelson.

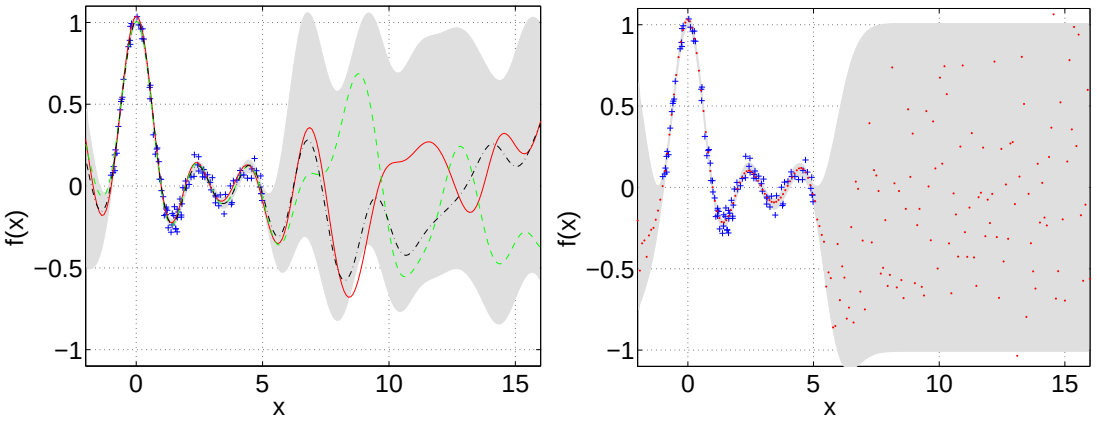
is denoted as $\overline{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$, so that the NMSE is normalized with respect to the MSE of a constant predictor. The number test samples is n_*.

The NMSE measures the accuracy of the predictive mean, ignoring the predictive variance. For traditional non-Bayesian regression methods, only this error measure was available. The MNLP, on the other hand, takes into account the predictive variance, weighting the error by the predicted degree of certainty.

2.4.1 One-dimensional toy problem

To illustrate the behavior of SSGP under extrapolation and evaluate the effect of its inherent periodicity, let us learn a simple noisy sinc function. We will generate 100 samples, for random values of $x \in [-1, 5]$, and add white Gaussian noise of variance $\sigma^2 = 0.05^2$. We will compare SSGP and SPGP using 20 spectral points and 40 pseudo-inputs respectively, sampled from the corresponding distributions. The remaining hyperparameters will be learned.

The predictive mean and a shaded area accounting for 95% of the (noise free) posterior probability are plotted in Fig. 2.2, for a very wide range of inputs.



(a) SSGP (b) SPGP

Figure 2.2: Three random samples of the joint posterior distribution obtained by (a) SSGP and (b) SPGP, when 100 noisy observations (plusses) of $\mathrm{sinc}(x)$ are provided. The shaded area accounts for 95% of the posterior (noise free) probability. Both methods use 40 basis functions.

SPGP provides an almost perfect match within the interpolation area and correctly predicts zero outside it. Since in the interpolation area data is dense enough, the predictive variance is approximately that of the injected noise, and we have verified that it remains almost constant at $\sigma_*^2(x) \approx \sigma^2 = 0.05^2$. In the extrapolation area, the variance grows and the correlation decays. This effect is clearly shown in Fig. 2.2.(b) for $x > 5$: The lack of correlation turns the posterior samples into white noise, so a cloud of dots is displayed.

SSGP performs equally well within the $[-1, 5]$ range where data is located. In fact, NMSE within that area for 600 test data points is 0.0022 for SSGP and 0.0023 for SPGP. The predictive variance in that range is also close to $\sigma^2 = 0.05^2$.

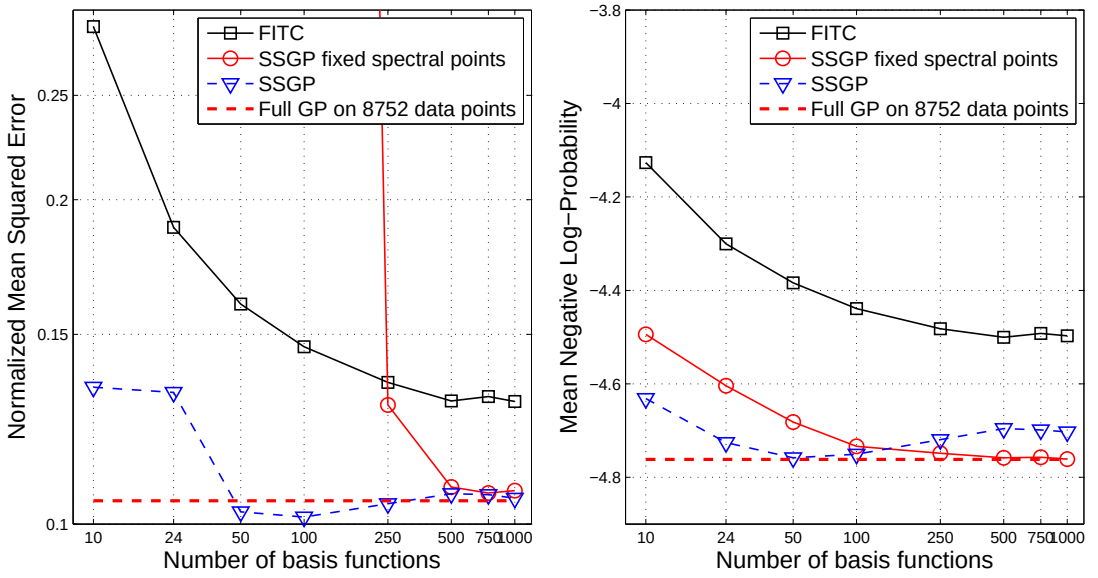
In the strong extrapolation area, for values of $x > 6$, the localized bases of SPGP correctly return to zero, whereas the periodic basis of SSGP do not. SSGP's undesirable behavior suggests we should avoid using it for strong extrapolation. Though in this case the incorrect predictive means are correctly flagged as unreliable by a corresponding growth in the predictive variance, for very big extrapolations, predictive variances are no longer trustworthy either.

2.4.2 *Elevators* and *Pole Telecomm* data sets

We will now experiment with some large,[4] real world data sets[5], previously used in other works such as Potgietera and Engelbrecht (2002, 2007); Torgo and da Costa (2000). Ten repetitions are run for each data set and number of basis functions. Average values of the performance measures are reported for each of the considered methods: SPGP, SSGP-fixed, SSGP, and full GP. We will run all methods using the general hyperparameter initialization proposed in Section 2.3.

The first problem is the *Elevators* data set, which has been obtained from the task of controlling an F16 aircraft. The target variable is related to the action taken on the elevators of the aircraft. After removing some constant inputs, it has 17 dimensions. We use the original split with 8752 data for training and 7847 for testing. Results are displayed in Fig. 2.3.

[4] To be able to quote full GP performances, we have had to resort to a 16 cpu 64 GB RAM machine.
[5] Both have been taken from http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html.



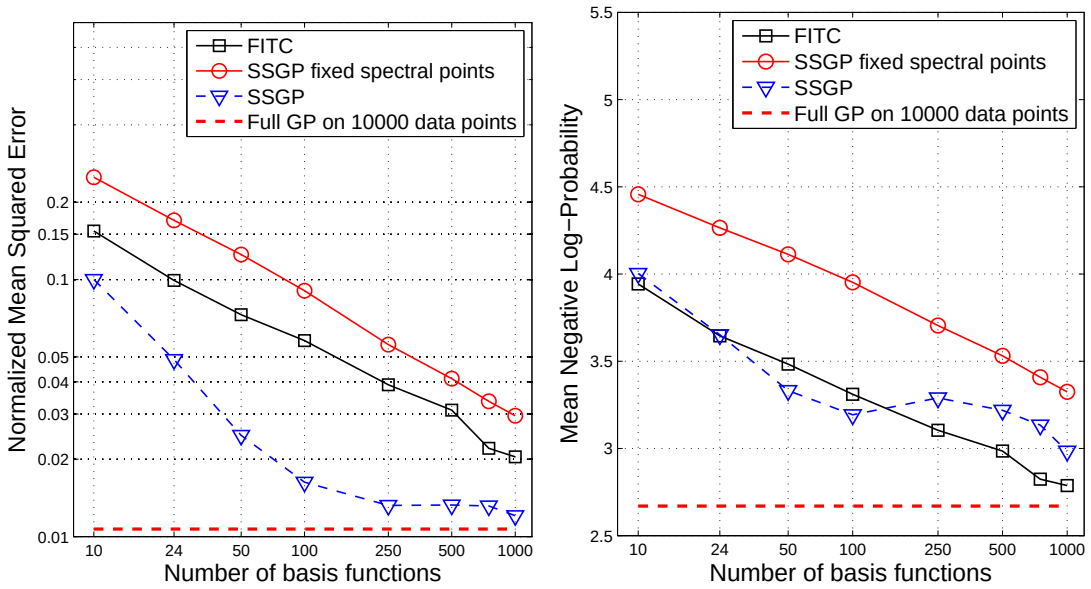
(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 2.3: NMSE and MNLP for SPGP, SSGP-fixed, SSGP and full GP for the *Elevators* problem.

SSGP performs much better than SPGP on this data set, both in terms of NMSE and MNLP. The data set is well represented with as few as 50 basis functions, with SSGP achieving a performance close to that of the full GP. It is interesting to notice how pouring many more basis than needed (up to 1000) into the model and fitting their frequencies does not produce any significant overfitting.

This data set shows an interesting effect for SSGP-fixed. Looking at the NMSE, it seems to perform very badly when the number of basis functions is low, quickly catching up when they are around 250-500. On the other hand, according to the MNLP, its performance seems to be quite good overall (significantly better than SPGP). It turns out that since the spectral points of SSGP-fixed cannot be adapted, on some runs the initial random disposition is not able to represent well the data set, thus resulting in a high NMSE that spoils the average over the ten runs. The model is correctly aware of its limitations, so for test points at unrepresented locations, large predictive variances are obtained, minimizing its impact on the MNLP. When the number of spectral points is big enough, it is less likely that big areas of the spectrum are left uncovered, so this effect disappears.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 2.4: NMSE and MNLP for SPGP, SSGP-fixed, SSGP and full GP for the *Pole Telecomm* problem.

The second data set, *Pole Telecomm*, is a commercial application described in Weiss and Indurkhya (1995), related to a telecommunications problem. After removing constant inputs, there are 26 input dimensions. Again, we use the original split, 10000 data for training and 5000 for testing. Upon inspection of the data set, we notice that both the inputs and the outputs take a discrete set of values. In particular, the outputs take values between 0 and 100, in multiples of 10. Though a multi-class classification method could be more suited to solve this problem, we will treat it as a regression problem. However, since the outputs are quantized, we are going to take this into account, lower bounding the value of σ^2 to the value of the quantization noise, which is $\text{bin}_{\text{spacing}}^2/12$. This lower bounding is applied to all the methods compared. The effect of the bound is to provide a slightly better estimate for σ^2 and therefore, better MNLP measures, but we have observed that this modification has no noticeable effect on NMSE values. Resulting plots are in Fig. 2.4.

SSGP performance is widely better in terms of NMSE. In terms of MNLP, both methods perform similarly, though SSGP seems to get worse as the number of basis functions grows. This shows how SSGP may sometimes produce wrong predictive

variances for a high number of spectral points.

SSGP-fixed improves steadily with the number of spectral samples, though does not have enough flexibility to compete with the other two methods.

2.4.3 *Kin-40k* and *Pumadyn-32nm* data sets

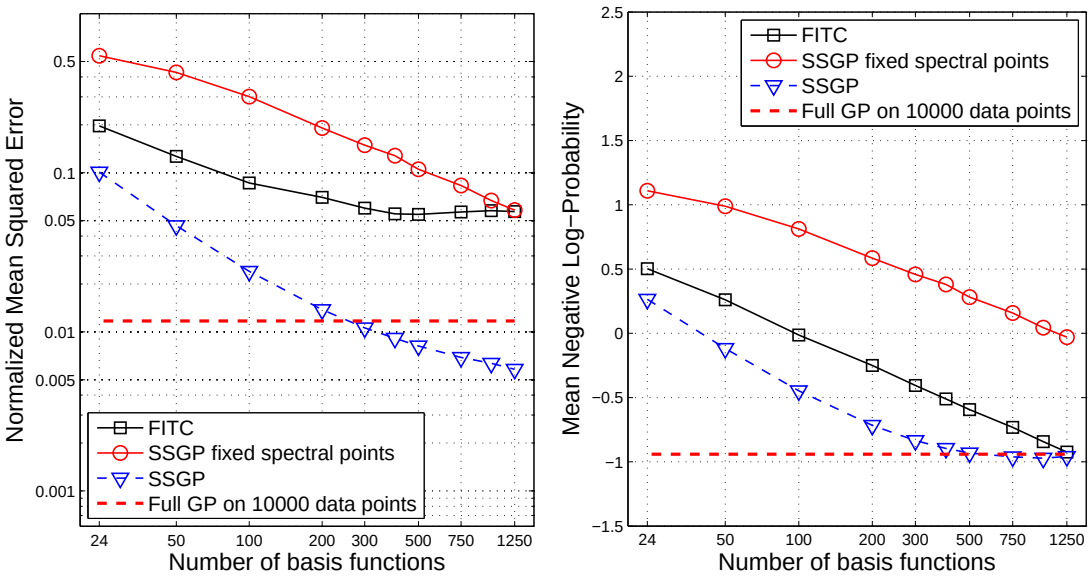
We now run SSGP on the same regression tasks presented in Seeger et al. (2003) and also used by Snelson and Ghahramani (2006), where SPGP was introduced. We follow precisely their preprocessing and testing[6] methods on the original splits. Each problem is run ten times and averaged. Data sets are *Kin-40k* (8 dimensions, 10000 training samples, 30000 testing) and *Pumadyn-32nm* (32 dimensions, 7168 training, 1024 testing), both artificially created using a robot arm simulator. They are highly non-linear and low noise.

Again, we will compare SPGP, SSGP-fixed, SSGP and a full GP. The sparse methods discussed in Seeger et al. (2003) are not included in the plots to avoid clutter, since SPGP already outperforms all of them. All hyperparameters are initialized as proposed in Section 2.3 unless stated otherwise.

Results for *Kin-40k* are shown in Fig. 2.5. SSGP performance on this data set is clearly superior to SPGP. Both data sets reach a tie in MNLP only for a very big number of basis functions. At that point, SSGP is still making much better predictions than SPGP (NMSE plot), but the noise power estimation is slightly low, making MNLP slightly worse. It is still remarkable the big amount of basis necessary to show a slight increase in the MNLP and that no overfitting is present in the NMSE.

Starting at 500 basis functions, SPGP's NMSE stops improving and even degrades slightly. On the other hand, SSGP not only keeps improving, but to our surprise, beats the full GP. This means that this particular regression task seems to be better represented by a set of trigonometric basis functions than it is by an (even infinite) set of localized squared exponentials. This might also help explain the much superior performance of SSGP compared to SPGP. Again, SSGP-fixed presents a modest performance, which improves steadily.

[6]For some reason, some previous works on these data sets provide an error measure that, together with their particular preprocessing, equals one half of the NMSE. We follow the same preprocessing but show NMSE values, so error measures coming from previous works have been adapted accordingly.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

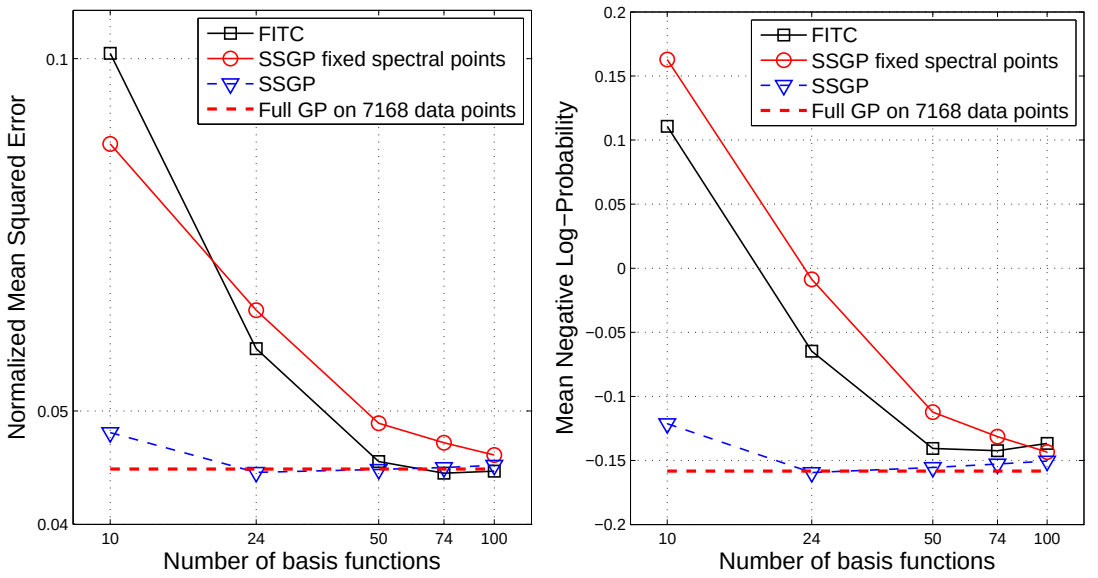
Figure 2.5: NMSE and MNLP for SPGP, SSGP-fixed, SSGP and full GP for the *Kin-40k* problem.

The *Pumadyn-32nm* problem can be seen as a test of the ARD capabilities of a regression model, since only 4 out of the 32 input dimensions are relevant. Following Snelson and Ghahramani (2006), to avoid the optimization process getting stuck at some high NLML value, length-scales are *initialized* from a full GP on a subset of 1024 training data points, for all compared methods. Then joint optimization over all hyperparameters is performed. Results are shown in Fig. 2.6.

All methods correctly perform the ARD, raising the length-scales of all inputs except $[4,\ 5,\ 15,\ 16]$, which are the ones really related to the prediction task. SSGP looks superior overall, achieving full GP performance with just 10 spectral points. It remains quite insensitive to overfitting when this number is increased.

2.4.4 *Pendulum* data set

So far we have seen data sets where SSGP was very competitive, producing excellent NMSE figures and reasonably good MNLP figures. Though this seems to be the common case, we can find tasks were SSGP fails to give proper predictive variances.



(a) NMSE (log-log plot)

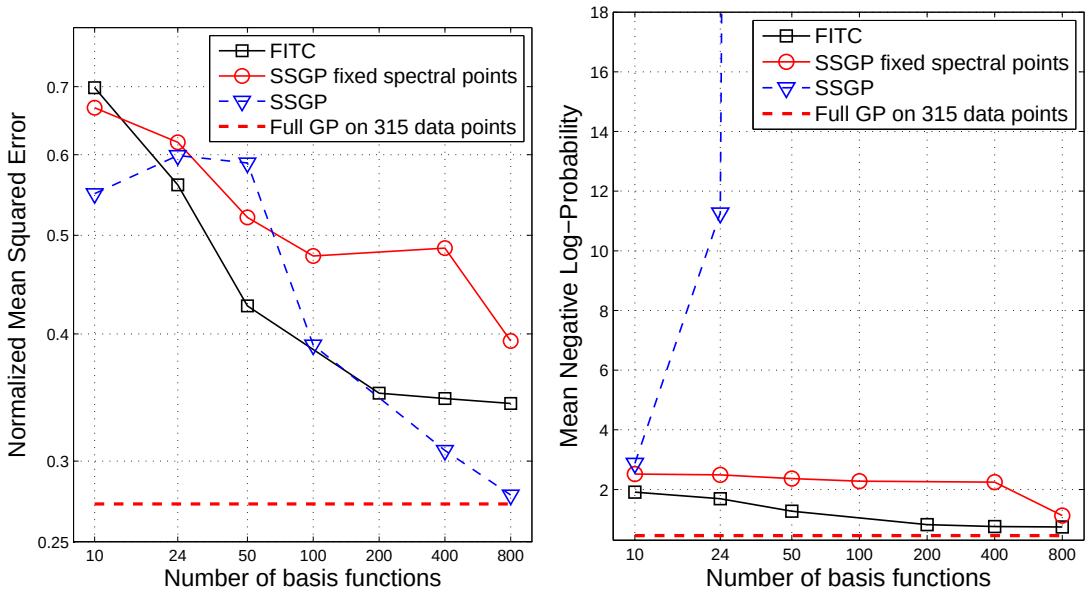
(b) MNLP (semilog plot)

Figure 2.6: NMSE and MNLP for SPGP, SSGP-fixed, SSGP and full GP for the *Pumadyn-32nm* problem.

The small data set *Pendulum* (9 dimensions, 315 training samples, 315 testing) represents the problem of predicting the change in angular velocity of a simulated mechanical pendulum over a short time frame (50 ms) as a function of various parameters of the dynamical system. The target variable depends heavily on all inputs. Fig. 2.7 shows results for the considered methods.

Results up to 800 basis functions are shown. This is for investigation purposes, since using more basis functions than samples[7] (315 in this case) presents no computational advantage with respect to a full GP. SSGP is shown to perform very badly on the MNLP measure. Unlike the NMSE, the MNLP takes the predictive variances into account penalizing more the errors in predictions with low variances (high certainty), see 2.21b. In this data set predictive means are quite accurate (as the NMSE plot shows) but predictive variances are exceedingly small, resulting in very bad MNLP figures. Wild disagreement among predictive distributions coming from different random initializations was also observed. It is interesting to note that the noise hyperparameter

[7] SPGP's pseudo-inputs are usually initialized to a random subset of the input samples. When $m > n$, all the input samples are used and additional $m - n$ pseudo-inputs are generated by a random convex combination of the existing inputs.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 2.7: NMSE and MNLP for SPGP, SSGP-fixed, SSGP and full GP for the *Pendulum* problem.

gets determined correctly, so it is not the cause of the small predictive variances.

This example shows that SSGP should be used with caution in tasks where predictive variances are of importance. We will show how to improve this type of behaviors in Chapter 3.

2.5 Overfitting versus overconfidence

A model is said to overfit when it is not only describing the underlying relationship between inputs and outputs in some data set, but also its inherent random noise. A model that overfits might seem good because it describes training data quite accurately (it has learned even part of its random fluctuations), but it is not: When tested on a separate data set with the same underlying relationship, results are much worse (since the random noise in the test data set is obviously different). In short, a model overfits when it learns random variations in training data and does not generalize well to unseen data.

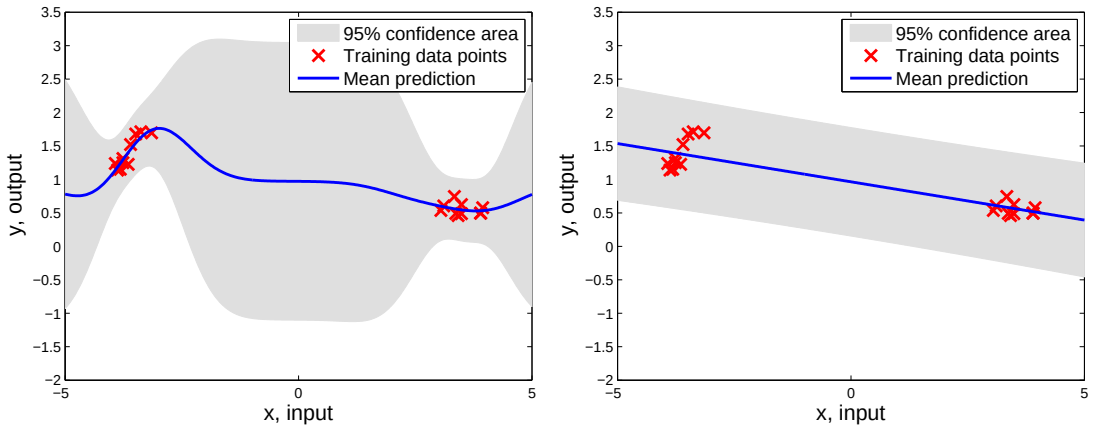
Overfitting typically appears when a model has a high expressive power or is excessively complex, i.e. it has too many degrees of freedom in relation to the amount of available data. When some generalization error measure (such as the test NMSE) is plotted versus the complexity of the model, overfitting can be easily spotted at the point at which models of higher complexity no longer reduce the error measure and instead increase it.

Bayesian models provide both an estimation of the output μ_* and an estimation of its uncertainty σ_*^2. There are cases, such as the *Pendulum* example in Section 2.4.4, in which as the complexity of a model grows, the generalization error measure (NMSE) decreases steadily, but the MNLP of the predicted values (which take the uncertainty estimation into account) increases. Thus, we can have a model which makes accurate predictions on unseen data but which is so overconfident that the actual data is unlikely under the predictive posterior.

Though overconfidence is sometimes regarded as plain overfitting, in this thesis we want to make a clear distinction between both, because whereas overfitting renders a model useless, an overconfident model can still be used successfully if we are only interested in mean predictions (by disregarding the uncertainty estimates).

Overfitting usually implies overconfidence, but the converse is not necessarily true. If a model overfits (in the sense given here of fitting noise), the noise power σ^2 is going to be underestimated, thus reducing the predictive variance and probably making the model overconfident too. If a model does not overfit, the noise component present in the uncertainty estimate is going to be large enough, but the projection of test data on seen examples $\mathbf{k_f}(\mathbf{x}_*)^\top(\mathbf{K_{ff}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{k_f}(\mathbf{x}_*)$ can nonetheless be overestimated. Overestimating this term in (1.15c) reduces the predictive variance and therefore an overconfident prediction can be made despite the absence of overfitting.

We will illustrate these ideas with a simple example. We draw a random function from a GP with one-dimensional ARD SE covariance function and sample it at 20 random points in the intervals $(-4, -3)$ and $(3, 4)$, leaving a relatively wide unsampled gap in $(-3, 3)$. Then we try to learn the underlying function from which those data points were drawn without any knowledge about the generating process. We use a full GP in which all hyperparameters are learned. Two possible covariance functions are considered: ARD SE (1.5) and linear ($k_{\text{Lin}}(\mathbf{x}, \mathbf{x}') = \sigma_0^2(1 + \mathbf{x}^\top\mathbf{x}')$). The posterior distribution is plotted in Fig. 2.8 for both cases.



(a) GP with ARD SE cov. fun.

(b) GP with linear cov. fun.

Figure 2.8: Shaded areas account for 95% of the posterior probability mass, given training data (crosses). Both panels correspond to full GP models where hyperparameters are learned via ML-II.

If we direct our attention to the area around $x = 0$, far away from data, the mean predictions from both models look equally reasonable. The uncertainty estimations, however, are clearly different. The non-linear GP in the left panel seems to be doing the right thing, almost reverting to the prior variance at $x = 0$, showing big uncertainty at a point which is far away from seen data. On the other hand, the linear model is much more confident about its predictions, because it is only considering the hypothesis space of linear models to determine the uncertainty. When tested on unseen data (which, as we know, comes from a non-linear model), the linear model would produce overconfident predictions yet no one would claim that the linear model is "overfitting data". If anything, it is underfitting it (the noise level estimate is bigger for the linear model than for the non-linear one).

Generalized linear models, such as SSGP, only consider a reduced hypothesis space with finitely many basis functions (a linear GP would be an extreme case in which only D bases are considered). Full ARD SE GPs, on the other hand, consider infinitely many basis functions. Thus, for a given data set of unknown nature, the former models, seeing less possible explanations for data, will (incorrectly) be more confident about their own predictions. A set of basis functions that describes well the underlying transformation for some data set but does not include enough alternative interpretations is going to produce overconfident predictions, yet not overfit, at least in the sense

described here.

If the set of basis functions is independent from data (such as in SSGP-fixed), the quality of the uncertainty estimates will grow as more bases are considered, since more alternative explanations are included. This is not necessarily the case when the bases are selected through ML-II, since they will try to better match available data, possibly resulting in overconfidence and, if enough basis functions are used, in overfitting.

Though SSGP may occasionally result overconfident, it rarely overfits data (it never did in the experiments shown in the previous section), provided that the number of spectral points is kept low in relation to the number of data points. If we want to make sure SSGP does not overfit, even for a large number of spectral points, we can do so by using a technique described in Section 3.2.1. Note that this technique does not protect against overconfidence.

Unlike SSGP, SPGP is an entirely different approximation that can not be interpreted as a generalized linear model. Instead, it uses a set of inducing variables as a bottleneck to make inference about data. The predictive uncertainties derived from this process are also considered in its predictions, so it hardly ever suffers from overfitting or overconfidence.

2.6 On the effect of learning the phases

In Section 2.1.2.2 we showed that SSGP can be regarded as a linear combination of cosines with some amplitude and phase, where both the amplitudes and the phases were integrated out. In this section, we investigate the benefits of integrating out the phases, by comparing with a very similar model where phases are regarded as deterministic hyperparameters instead.

The proposed deterministic-phase model is analogous in structure to SSGP

$$f_{\text{MCN}}(\mathbf{x}) \;=\; \sum_{i=1}^{m} c_i \cos(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i) \;, \tag{2.22}$$

but $\{\varphi_i\}_{i=1}^m$ are now regarded as hyperparameters. Additionally, for analytical tractability, we will also replace the symmetric-Rayleigh prior placed on $\{c_i\}_{i=1}^m$ with a matching-moments Gaussian $p(c_i) = \mathcal{N}(c_i|0, 2\sigma_0^2/m)$.

We will refer to this deterministic-phase model as Marginalized Cosine Network (MCN), since it is a analogous to a Neural Network with cosine activation[8], where the output weights have been marginalized out. We will devote Chapter 3 to the study and improvement of general Marginalized Networks and will use MCNs as our running example. MCNs can be considered as a cosines-only approximation to SSGP.

The only two differences between MCNs and SSGP are that they use different priors on the weights and that SSGP integrates out phases whereas MCNs do not. We will now argue that the first difference is almost irrelevant in practice.

The prior on the weights only affects the model through the effect it produces on the prior on $\mathbf{f}$. This latter prior is a linear combination of the priors on the weights[9], so $\mathbf{y}$ will end up having an almost-Gaussian density if the number of independent weights is big enough, due to the Central Limit theorem. Replacing the symmetric-Rayleigh prior on the weights with a matching-moments Gaussian replaces that almost-Gaussian density on $\mathbf{f}$ with an actual Gaussian of matching-moments. The effect of this change is negligible, even for a small number of weights.

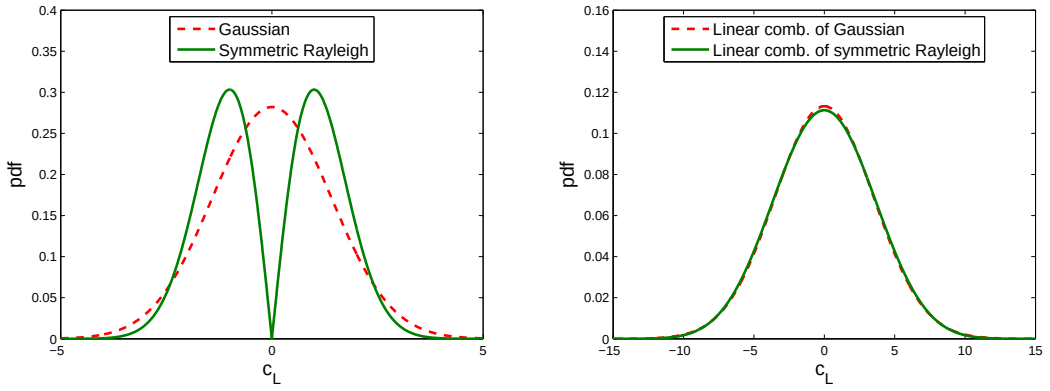
To see that the number of symmetric-Rayleigh weights that need to be combined to achieve an almost-Gaussian density is very low, we provide Fig. 2.9. In panel (a), we show a symmetric-Rayleigh distribution and a matching-moments Gaussian. In panel (b), we show the density of a linear combination of just ten independent weights, for both densities. Recall that linearly combining random variables amounts to scaling and *convolving* their densities, hence a linear combination of independent symmetric-Rayleigh distributed weights does not have a density with zero mass around zero, despite all independent weights having it.

The second difference between SPGP and MCN, regarding the phases as deterministic hyperparameters instead of integrating them out, has deeper implications: The stationarity property is lost. In particular, the point wise variance of SSGP was constant (σ_0^2), but in MCN, it is input-dependent

$$\mathbb{V}[f_{\text{MCN}}(\mathbf{x})] \;=\; \sum_{i=1}^{m} \mathbb{V}[c_i] \cos^2(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i) = \frac{2\sigma_0^2}{m} \sum_{i=1}^{m} \cos^2(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i) \;,$$

[8]To be more precise, to a Multi-Layer Perceptron (MLP) architecture with a single hidden layer, using cosine-type activation for the hidden layer and linear activation for the output layer.

[9]Recall that $\mathbf{f} = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{x})$ is deterministic.



(a) pdf of independent variables

(b) pdf of linear combination

Figure 2.9: (a) Symmetric Rayleigh pdf and Gaussian pdf, both with the same mean and variance. (b) Resulting pdf of the linear combination of 10 independent random variables with given pdfs.

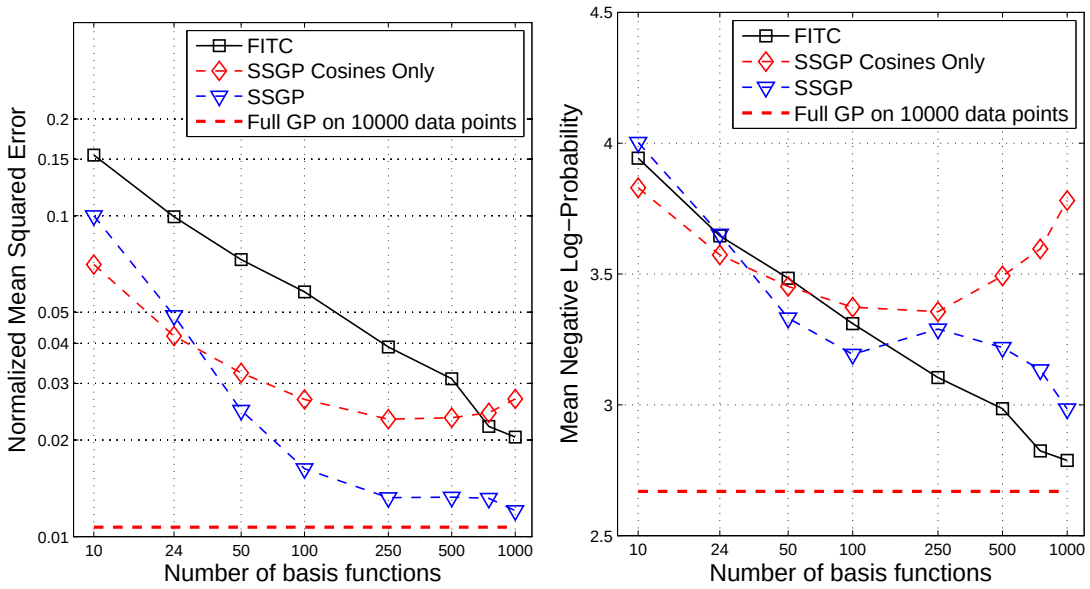
and takes values in the range $[0, 2\sigma_0^2]$. For the ranges of the input domain where the prior variance takes near-zero values, the predictions may be overconfident (since the predictive variance can only be smaller than the prior variance). This problem is similar to that of the SR approximation, discussed in Section 1.2.3.

Perhaps a simpler way to illustrate the close relation between SSGP and MCN is by inspecting the effective covariance function of the latter

$$k_{\text{MCN}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_0^2}{m} \sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top (\mathbf{x} - \mathbf{x}')) + \frac{\sigma_0^2}{m} \sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top (\mathbf{x} + \mathbf{x}') - 2\varphi_i) \ ,$$

and noticing that its first term is precisely the SSGP covariance function (2.7), whereas the second term adds a non-stationary perturbation. Since the second term is an average of cosines, if the phases $\{\varphi_i\}_{i=1}^m$ are uniformly distributed, the perturbation vanishes as more basis functions are added. Further details on the infinite-limit convergence and the derivation of this covariance function can be found in Section C.3 of Appendix C.

The derivation of the predictive equations for the MCN model (see Section 3.1.2 and (3.4) for details) is almost identical to that of SSGP. Model selection is also performed similarly, by jointly optimizing the NLML wrt all hyperparameters (which now include the phases). Phases are initialized randomly from a uniform density in the interval $[0 \dots 2\pi]$. The detailed procedure is provided in Section 3.1.2.1. One advantage



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 2.10: NMSE and MNLP for SPGP, MCN, SSGP and full GP for the *Pole Telecomm* problem.

of the MCN model is that, for a fixed number of frequencies h, the design matrix consists only of h columns (as opposed to the $2h$ columns of SSGP). That roughly halves storage costs and divides computational costs by four. Conversely, for a fixed number of basis functions, MCN uses twice as many frequencies as SSGP.

Despite this advantage, MCN is less accurate than SSGP in the data sets discussed in the experimental section. As an example, we show its performance on the *Pole-Telecomm* data set in Fig. 2.10 (keeping, as in Section 2.4.2, $\sigma^2 > \text{bin}^2_{\text{spacing}}/12$). We can observe how MCN starts overfitting at $m = 500$ basis functions (both error measures start degrading) whereas SSGP keeps improving. This shows the benefit of phase integration, which is the fundamental difference among both methods.

2.7 Summary and conclusions

In this chapter we have introduced Sparse Spectrum GPs (SSGP), a sparse GP model that can be regarded as a linear combination of trigonometric basis functions where

both weights and phases have been integrated out. Hyperparameters are jointly selected by minimizing the NLML of the model, which can be done in a very efficient manner. This model presents the following properties:

- It has no location parameters.
- It is stationary.
- It has a deep connection with stationary full GPs: It can approximate any stationary full GP by making the spectral points follow the distribution of the spectral density of the full GP's covariance. The approximation becomes exact as the number of spectral points tends to infinity.

The performance and behavior of the new model has been tested extensively. Learning the spectral points results, in most cases, in significant improvement over the current state of the art, SPGP. Fixing the spectral points yields a less accurate method, but one that (unlike SPGP) is known to converge to the full GP in the infinite limit, and for which only a very small number of hyperparameters needs to be selected.

Two potential problems with SSGP (only present when the spectral points are *learned*) have been identified:

- SSGP may overfit if too many spectral points (in relation to the number of available data samples) are used. If this happens, noise power is typically underestimated and the predictive means and variances degrade. This problem is hardly ever the case (never was in the experiments). A simple workaround for this type of problem will be described in Section 3.2.
- SSGP may be overconfident. On some regression tasks, even when predictive means and noise power are properly determined, the predictive variances can be underestimated due to the limited hypothesis space of the model, as discussed in Section 2.5. This problem appeared only in the *Pendulum* data set, and is only relevant if we are interested in uncertainty estimates. A possible workaround will be presented in Section 3.3.

Despite these potential problems, SSGP (both with fixed and selectable spectral points) seems to be a promising approach for tackling large-scale regression problems that require both efficiency and accuracy.

Chapter 3

Marginalized Networks

In this chapter we will introduce Marginalized Networks (MNs), which generalize the SSGP model described in Chapter 2. MNs are generalized linear regression models that encompass traditional network topologies such as Multi-Layer Perceptrons or Radial Basis Functions Networks. However, unlike traditional networks, MNs regard the weights of the output layer as random variables which can be marginalized out instead of learned. Although this gives MNs some advantages over their classical counterparts, some problems, such as overfitting, are still present. In order to overcome these problems and develop practically useful methods for large-scale learning, we propose two new methods: A simple technique, called "noise bounding", and a more sophisticated one called "network mixing". Both of them run within the same complexity order as previously proposed approaches, and improve on the accuracy of SPGP and even SSGP.

This chapter is organized as follows: In Section 3.1 we formalize the concept of Marginalized Networks and describe their associated drawbacks; in Section 3.2 we introduce noise bounding and provide experiments to show its usefulness, including comparisons with SPGP and SSGP; in Section 3.3 we introduce network mixing and provide analogous experiments; in Section 3.4 a scheme for supervised dimensionality reduction using MNs is described; finally, Section 3.5 summarizes and concludes the chapter.

3.1 The Marginalized Network (MN) model

Consider the generalized linear model

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{m} w_i \phi(\mathbf{u}_i, \mathbf{x})\,, \tag{3.1}$$

where $\{\phi(\mathbf{u}, \cdot)\}_{i=1}^{m}$ is a family of scalar basis functions parametrized by $\{\mathbf{u}_i\}_{i=1}^{m}$ (the so-called "input weights") and $\{w_i\}_{i=1}^{m}$ is a set of weights that control the scaling of each basis function (the "output weights"). This model represents an arbitrary nonlinear mapping between samples $\mathbf{x} \in \mathbb{R}^D$ in a multidimensional input space and some scalar output space.

Many classical regression methods can be represented by this model, notably Radial Basis Functions Networks (RBFNs) and Multi-Layer Perceptrons (MLPs) with a single hidden layer, both of which are a type of Neural Networks (NNs). Networks of this type are usually trained by selecting their weights $\{w_i, \mathbf{u}_i\}_{i=1}^{m}$ so as to minimize the squared error over the training set, i.e. $\sum_{j=1}^{n}(f(\mathbf{x_j}) - y_j)^2$. This results in a number of problems:

- Overfitting: When m is large, the model is too flexible and it fits the data set almost perfectly, as if no noise was present in the observations. This results in bad generalization and poor predictions on new data. Thus m is not only used to control the amount of computation we are willing to spend on some regression task, but it is also a critical parameter that we must select carefully: Larger values of m, though devoting more computing power, may result in less performance.

- Non-convex cost functional (with respect to the input weights): The squared error cost functional is not convex for this type of networks, so convergence to poor local minima is possible.

- Cross-validation must be used: The squared error cost functional is not appropriate to select other sensitive parameters (such as m or the amount of weight decay in MLPs), so these parameters must be selected through cross-validation. This has some drawbacks: Only a discrete set of values for the parameters is considered, computational cost scales exponentially with the number of parameters, and validation sets are required, thus reducing the number of samples that can effectively be used during model selection.

- Deterministic predictions: Estimates of the outputs for new data points are provided without a hint about its uncertainty. Also, no estimation of signal-to-noise ratio in the data set under consideration is provided.

On the positive side, they are very cheap to train, taking only $O(mn)$ time per epoch[1] for simple, gradient-based optimizations algorithms and $O(m^2n)$ time for more sophisticated algorithms such as Levenberg-Marquardt[2]. Several epochs must of course be run until convergence is achieved.

A possible solution to the aforementioned problems is to use a non-parametric, Bayesian approach: Use an infinite number of basis functions, place a prior on all the weights of the network, and integrate them out. Difficult as it seems, it turns out that for some sets of basis functions, it is possible to do this analytically, as proved in Williams (1997). The resulting model in these cases is precisely a full GP, for which, as we know, it is possible to make inference with a finite (but huge) amount of computation.

Instead of integrating out all the parameters of the network and then sparsify the resulting model, in this chapter we propose to enforce sparsity by design, integrating out only the *output* weights. We will call this model Marginalized Network (MN).

3.1.1 Definition

We define a Marginalized Network as a Bayesian regression model following these equations:

$$y = f(\mathbf{x}) + \varepsilon \qquad f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) \tag{3.2a}$$

$$p(\varepsilon) = \mathcal{N}(0, \sigma^2) \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \sigma_0^2/(m\sigma_\mathrm{p}^2)\mathbf{I}_m), \tag{3.2b}$$

where $\mathbf{w} = [w_1, \ldots, w_m]^\top$ and $\boldsymbol{\phi}(\mathbf{x}) = [\phi(\mathbf{u}_1, \mathbf{x}), \ldots, \phi(\mathbf{u}_m, \mathbf{x})]^\top$. Values σ_0^2, σ^2 and $\{\mathbf{u}_i\}_{i=1}^m$ are regarded as hyperparameters and σ_p^2 is a normalizing constant to ensure that σ_0^2 properly represents the average signal power $\mathbb{E}[f^2(\mathbf{x})]$. Constant σ_p^2 is

[1] In the context of NNs, an epoch refers to a complete sweep of the optimization process over all data samples.

[2] When quoting the computational cost of our algorithms, we usually omit the dependence on D, which is linear. For Levenberg-Marquardt, though, this dependence is quadratic. Thus Levenberg-Marquardt is computationally more expensive than the algorithms presented in this thesis.

just the average power of the basis functions over the input space, so it is known a priori.

Design matrix $\mathbf{\Phi_f} = [\boldsymbol{\phi}(\mathbf{x}_1),\ \ldots,\ \boldsymbol{\phi}(\mathbf{x}_n)]$ of the training set can be used to express the joint distribution of the latent values compactly:

$$p_{\text{MN}}(\mathbf{f}|\{\mathbf{x}_j\}_{j=1}^n) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \sigma_0^2/(m\sigma_{\text{p}}^2)\mathbf{\Phi_f}^\top\mathbf{\Phi_f}),$$

which is just a GP with the following (noiseless) covariance function

$$k_{\text{MN}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_0^2}{m\sigma_{\text{p}}^2}\boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\phi}(\mathbf{x}'). \tag{3.3}$$

The definition of MN corresponds to a sparse GP. When the number of basis functions is infinite and the input weights are selected according to some distribution, a full GP arises. This distribution, together with the functional form of $\boldsymbol{\phi}(\cdot,\cdot)$, determines whether the covariance of the full GP can be computed analytically. When working with full GPs, only combinations of distributions and functional form that yield analytical covariance functions can be used. On the contrary, when considering MNs, any combination can be used.

We can use all the equations from the GP framework just by replacing $\mathbf{K_{ff}}$ with the low rank matrix $\frac{\sigma_0^2}{m\sigma_{\text{p}}^2}\mathbf{\Phi_f}^\top\mathbf{\Phi_f}$. Suitable values for the input weights and the signal and noise hyperparameters (σ_0^2 and σ^2) are found by jointly optimizing the NLML (1.18).

Since the covariance matrix of this GP is low rank by construction, previous GP equations can be re-stated in a computationally cheaper form. Thus, for making predictions at new test points, (1.15) becomes:

$$p_{\text{MN}}(y_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(y_*|\mu_{\text{MN}^*}, \sigma^2_{\text{MN}^*}) \tag{3.4a}$$

$$\mu_{\text{MN}^*} = \boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y} \tag{3.4b}$$

$$\sigma^2_{\text{MN}^*} = \sigma^2 + \sigma^2\boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_*) \tag{3.4c}$$

where $\mathbf{A} = \mathbf{\Phi_f}\mathbf{\Phi_f}^\top + \frac{m\sigma_{\text{p}}^2\sigma^2}{\sigma_0^2}\mathbf{I}_m$ is an $m \times m$ matrix, much smaller than the covariance matrix. To select select free parameters through type II Maximum Likelihood (ML-II), (1.18) can be expressed in the following computationally-efficient form:

$$\begin{aligned} -\log p_{\text{MN}}(\mathbf{y}|\theta) = &\left[\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{\Phi_f}^\top\mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y}\right]/(2\sigma^2) \\ &+ \frac{1}{2}\log|\mathbf{A}| - \frac{m}{2}\log\frac{m\sigma_{\text{p}}^2\sigma^2}{\sigma_0^2} + \frac{n}{2}\log 2\pi\sigma^2. \end{aligned} \tag{3.5}$$

Equations (3.4) and (3.5) can be computed in $\mathcal{O}(m^2 n)$, the same bound offered by SPGP. Actually, for the same number of basis functions, MNs are faster because the low rank decomposition of the covariance matrix is simpler. Further computational savings and numerical accuracy can be achieved by using the Cholesky decomposition, as detailed in Section D.3 of Appendix D. The largest matrix in this computations is the design matrix $\mathbf{\Phi_f}$, so only $\mathcal{O}(nm)$ storage space will be needed.

It is interesting to notice how the predictive mean can be expressed as the output of a traditional network, i.e. in the format of (3.1), using the selected input weights and $\mathbf{w} = \mathbf{A}^{-1}\mathbf{\Phi_f}\mathbf{y}$ as output weights. Unlike SPGP, MNs preserve the structure of traditional networks, so they can be used to train RBFNs, MLPs, etc.

3.1.2 Marginalized Cosine Networks (MCN)

Choosing the family of basis functions

$$\phi(\mathbf{u}, \mathbf{x}) = \cos(\mathbf{u}^\top \tilde{\mathbf{x}})$$

parametrized by $\mathbf{u}$, where $\tilde{\mathbf{x}} = [1,\ \mathbf{x}^\top]^\top$ is the augmented input vector, and plugging it into (3.2) retrieves the Marginalized Cosine Network mentioned in Section 2.6.

Constant σ_{p}^2, needed for (3.4) and (3.5), can be derived from the family of basis functions

$$\sigma_{\text{p}}^2 = \lim_{L\to\infty} \frac{1}{L^D} \int_{C_L} \cos^2(\mathbf{u}_i^\top \tilde{\mathbf{x}}) \mathrm{d}\mathbf{x} = \frac{1}{2},$$

where the region C_L is a cube of edge L, centered at the coordinate origin.

3.1.2.1 Model selection

We will describe model selection for MCNs, but the procedure for other types of MNs is analogous. MCNs are trained by minimizing the Negative Log Marginal Likelihood (NLML) of the model, as described by (3.5). Each input weight $\mathbf{u}_i$ is in turn parametrized as follows: $\mathbf{u}_i = [\varphi_i,\ (\mathbf{L}^{-1}\boldsymbol{\omega}_i)^\top]^\top$, where $\mathbf{L}$ is a diagonal matrix of size $D \times D$ whose elements are the length-scales $\ell_1 \ldots \ell_D$. With this definition, to completely define a model we must specify σ_0^2, σ^2, $\{\ell\}_{d=1}^D$, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$.

MCN with selectable input weights First, we will consider the case in which all the aforementioned hyperparameters are learned. Selecting both $\{\ell_d\}_{d=1}^D$ and $\{\boldsymbol{\omega}_i\}_{j=1}^m$ effectively overparametrizes the model, but as we discussed in Section 2.3 for SSGP, this eases the optimization process and enables ARD to be correctly performed. The detailed model selection procedure is:

1. Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, and σ^2 to some sensible values. (We will use: One half of the ranges of the input dimensions, the variance of the outputs $\{y_j\}_{j=1}^n$ and $\sigma_0^2/4$, respectively).

2. Initialize $\{\boldsymbol{\omega}\}_{i=1}^m$ from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_D)$ and $\{\varphi_i\}_{i=1}^m$ from a uniform distribution in $[0, 2\pi)$. This initially approximates the ARD SE covariance function (see below).

3. Minimize (3.5), the NLML of the model, wrt to $\{\ell\}_{d=1}^D$, σ_0^2, σ^2, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$. Since analytical derivatives are available, conjugate gradient descent can be used. Recall that for cosine basis, $\sigma_{\mathrm{p}}^2 = 1/2$.

As with SSGP, it is possible to compute all of the $D + 2 + m(D + 1)$ NLML derivatives in $\mathcal{O}(m^2 n)$ time. Details on how to do this in a numerically stable way are provided in Section E.2 of Appendix E. As in previous methods, storage space is dominated by the design matrix, so it is $\mathcal{O}(mn)$.

SPGP uses $D+2+mD$ hyperparameters, so for the same number of basis functions MCN and SPGP roughly need the same number of hyperparameters. The additional flexibility of MCNs with respect SPGP does not come from the $m - D$ additional degrees of freedom[3], but from the nature of the approximation, that is not constrained to approximate any concrete covariance function.

MCN with fixed input weights We know from Chapter 2 that if vectors $\{\mathbf{s}_i = (2\pi\mathbf{L})^{-1}\boldsymbol{\omega}_i\}_{i=1}^m$ are distributed according to the spectral density of some stationary covariance function, as m tends to infinity SSGP converges to a stationary full GP with that stationary covariance function. This also holds for MCN, provided that the phases are uniformly distributed in any $[n_1\pi, n_2\pi]$ range with $n_1, n_2 \in \mathbb{Z} : n_2 > n_1$. See convergence proofs in Sections C.2 and C.3 of Appendix C.

[3] Recall that D of MCN's hyperparameters are redundant

This explains the proposed initialization for $\{\boldsymbol{\omega}\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$: If we draw $\{\boldsymbol{\omega}_i\}_{i=1}^m$ from a $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_D)$ distribution, then $\{\mathbf{s}_i = (2\pi\mathbf{L})^{-1}\boldsymbol{\omega}_i\}_{i=1}^m$ will be distributed as $\mathcal{N}(\mathbf{s}|\mathbf{0}, (2\pi\mathbf{L})^{-1})$, which in turn corresponds to the ARD SE covariance function, as per Section 2.1.3. Thus, for finite m, the model will be initialized to approximate a GP with the ARD SE covariance.

If we let $\{\varphi_i, \boldsymbol{\omega}_i\}_{i=1}^m$ fixed during the third step of the process, the model keeps being an approximation to the full GP, and it has roughly the same properties as SSGP-fixed (though it is not stationary). By only fixing $\{\boldsymbol{\omega}_i\}_{i=1}^m$ we retain most of the properties of SSGP-fixed, though strict convergence to the full-GP is lost. We will refer to this latter model as MCN-fixed. As opposed to MCN, MCN-fixed:

(a) barely overfits, since just $D + m + 2$ hyperparameters are selected, (in contrast with the $D + 2 + (D + 1)m$ hyperparameters that need to be selected for MCN);

(b) can be trained much faster, for the same reason; and

(c) needs more basis functions to fit data, since input weights are fixed (up to a common scaling).

3.1.2.2 SSGP as an MCN

SSGP can be obtained as an instantiation of MCN by additionally constraining input weight vectors $\{\mathbf{u}_i\}_{i=1}^m$ to have paired form $\{[0,\ 2\pi\mathbf{s}_r^\top]^\top, [-\pi/2,\ 2\pi\mathbf{s}_r^\top]^\top\}_{r=1}^h$. For h spectral points, the total number of basis functions is therefore $m = 2h$. As we showed in Section 2.1.2.2, this constraint has the effect of integrating out the phase, thus making the model more robust to overfitting and having slightly less hyperparameters. The improved robustness of SSGP over MCN was shown in Section 2.6.

3.1.3 Drawbacks of MNs

MNs achieve to some extent the advantages of full GPs (probabilistic predictions, ML-II model selection, and some overfitting resistance) while still being a computationally tractable model for big data sets. If the input weights are fixed, they can be safely used, without any risk of overfitting, but in most cases, this does not yield state-of-the-art performance. If the input weights are selected by ML-II, higher performance is achieved, but new problems appear:

1. When the number of basis functions m is large, the quality of the predictive mean may deteriorate, i.e. the model may overfit. Even though this effect is reduced by integrating out the output weights, selecting the input weights makes the model flexible enough so as to fit part of the noise present in the data set.

2. If m is large, the quality of the predictive variance may deteriorate too, usually resulting in an underestimation of the uncertainty level. This problem may appear even when predictive means are accurately estimated, (the overconfidence problem, see Section 2.5).

3. ML-II model selection is possible for MNs (there is no need to resort to cross-validation), but the dimension of the search space is large (such as that of traditional networks). Convergence to local minima is possible.

Problem 3 is typical of most sparse GP algorithms (including SPGP): Selecting a sparse set of basis functions to fit the data set is almost never a convex problem and thus can suffer from local minima. Fortunately, in most cases, local minima are good enough solutions. In the rare case in which this problem appears, there are two possible workarounds: a) restart the algorithm with a different initialization, hoping for a better solution to be reached; and if this does not work, b) try a different family of basis functions, since this will change the search space itself. Solution b) usually implies also a change the covariance function that is being approximated. However, in Chapter 4, we will present a variation of SPGP that allows to maintain the target covariance function while changing the family of basis functions.

If we are only interested in mean predictions (as it is always the case for people using traditional regression networks), we can disregard problem 2. In that case, problem 1 can be solved in a simple and efficient manner as we detail in Section 3.2,

If we are also interested in full probabilistic predictions, we need to reduce the effect of the limited hypothesis space. This can be achieved by mixing several networks. We detail this procedure in Section 3.3. Though for big enough m predictive variances may still be underestimated, mixing drastically reduces overconfidence.

Of course, for very sparse models (small m), problems 1 and 2 do not apply, so good results are expected even if we use plain MNs. Since MNs are improved versions of classical networks, they are also amenable to classical solutions for overfitting (such as using cross-validation to select a suitable, small enough, value for m).

3.2 Bounded-Noise Marginalized Networks (BN-MN)

In this section we introduce Bounded-Noise Marginalized Networks (BN-MNs), a simple yet effective improvement over MNs, that prevents predictive means from degrading as the number number of basis functions m grows. This renders BN-MNs an efficient tool for those problems in which point estimates are the main interest. Predictive variances are also improved (and often become reasonably good, even for large m), but are not completely reliable, as we show in the experimental section. We will see in Section 3.3 a method which is better suited for problems where accurate predictive variances are needed.

3.2.1 Noise bounding

The probabilistic regression models that we are using are always of the form $y = f(\mathbf{x}) + \varepsilon$, i.e., observation y consists of a latent function $f(\mathbf{x})$ (the desired signal) plus Gaussian noise ε. We use hyperparameter σ_0^2 to denote the average power of $f(\mathbf{x})$, whereas σ^2 denotes the noise power. Both hyperparameters are unknown and are selected so as to maximize the evidence of the model for a given data set. After the model selection procedure converges and we have estimates of σ_0^2 and σ^2, it is reasonable to expect the empirical power of the observations $\frac{1}{n}\sum_{j=1}^{n} y_j^2$ to be roughly equal to the average power of the model, $\sigma_0^2 + \sigma^2$. Thus, model selection is used to determine how the power of the observations is split between signal and noise. This values can be used to obtain a Signal-to-Noise Ratio (SNR) estimate for a given data set.

When the number of basis functions m in an MN is too small, only a coarse fit to available data can be provided, due to the lack of expressive power. As m grows, the fit improves and, when testing on a separate set of data, the NMSE decreases. However, if we keep increasing m, at some point the number of basis functions will be large enough so as to fit not only signal $f(\mathbf{x})$, but also the observation noise. This still decreases the training NMSE, but the test NMSE gets increased, since noise in both sets is not correlated. This undesirable noise-fitting effect is known as overfitting.

A traditional approach to avoid overfitting is to limit the expressive power of the model. In this case, we would have to select a value for m (e.g., using cross-validation) that is large enough so as to fit available data whereas it is small enough so as not to

start fitting noise. This approach has two clear drawbacks: Choosing which values of m are going to be swept during cross-validation is not a trivial problem. Also, assuming that we somehow know the M potential values of m that we are going to consider, the model selection procedure must be run MN_{folds} times, where N_{folds} is the number cross-validation folds (5 is a typical value). The overall multiplicative factor can be very big, thus defeating the purpose of a sparse, fast method.

When overfitting appears in an MN, at least some part of the noise is not being identified as such, and $f(\mathbf{x})$ is fitting both the signal and that part of the noise. In such a model, σ^2 is underestimated (since less noise is needed to account for the deviations of the observations from $f(\mathbf{x})$) and σ_0^2 overestimated (since more power is needed to account for both the signal and the part of the noise that is identified as signal). Thus, some of the power that should be identified as noise gets identified as signal, but $\sigma_0^2+\sigma^2$ keeps being roughly equal to the average power of the observations.

In this context, we reason as follows: If we knew beforehand a good approximation σ^2_{min} of the true noise power and only allowed model selection to search in the interval $\sigma^2 \in [\sigma^2_{\text{min}}, \infty)$, noise could not be underestimated. Thus σ_0^2 could not be significantly overestimated (since $\sigma_0^2 + \sigma^2$ is roughly constant for each given data set) and $f(\mathbf{x})$ should only fit the signal (since noise is already fitted). We call this idea "noise bounding", and MNs using it Bounded-Noise Marginalized Networks (BN-MNs).

The reason for the noise power σ^2 to be lower-bounded by an estimation σ^2_{min} instead of just fixing it to the estimation is twofold: First, during model selection, the easiest path between the starting model and the desired model might traverse higher noise parts of the model space, so even if σ^2 takes the value σ^2_{min} both at the starting and finishing value, letting it take higher values might ease optimization. Second, if $f(\mathbf{x})$ underfits data for some reason, the value of σ^2 must be higher, to compensate for the lack of fit.

It is also possible to give another argument about why lower-bounding σ^2 provides protection against overfitting for the predictive mean. The predictive mean of an MN is:

$$\mu_{\text{MN}^*} = \boldsymbol{\phi}(\mathbf{x}_*)^\top \left(\boldsymbol{\Phi}_\mathbf{f} \boldsymbol{\Phi}_\mathbf{f}^\top + \tfrac{m\sigma_\text{p}^2\sigma^2}{\sigma_0^2} \mathbf{I}_m \right)^{-1} \boldsymbol{\Phi}_\mathbf{f} \mathbf{y},$$

which has the form of regularized linear regression in the feature space induced by the basis functions. The regularization factor is $\frac{m\sigma_\text{p}^2\sigma^2}{\sigma_0^2}$. Thus, by lower bounding σ^2 we

are additionally imposing a lower bound on the regularization level[4]. Regularization is known to avoid overfitting, and assuming again that some noise power estimation σ^2_{min} is available, we can estimate the regularization factor.

An interesting observation follows: The optimal regularization constant for any network following (3.1) (notably including classical networks such as RBFNs and MLPs) is inversely proportional to the SNR of the observations, with known proportionality constant $m\sigma^2_{\text{p}}$.

3.2.2 Obtaining a noise power estimate

To effectively apply this idea to train a BN-MN we also need a reasonably accurate estimation σ^2_{min} of the noise power. A simple method to achieve this is to run model selection on an MN of the appropriate size (same m as the BN-MN), with the input weights selected from some distribution and fixed (i.e., MN-fixed[5]), then take the selected value of σ^2 as σ^2_{min}. Since the basis functions are fixed, the model cannot overfit the data. Thus, the selected σ^2_{min} would be, if anything, above the true noise level.

In the concrete cases of SSGP and MCN, we already know that fixing the input weights (also known as spectral points in the SSGP model) to be distributed according to the normalized spectral density of some stationary covariance function provides an approximation which tends to a full GP with that stationary covariance function as m tends to infinity. This convergence, as well as the absence of overfitting was experimentally shown in Section 2.4. For other types of MNs with different activation function, there may or may not be analytical expressions for the covariance function achieved in the infinite limit, but they will nonetheless converge to a full GP and present no overfitting.

When m is small, only a few fixed basis functions are used, so data is probably underfit. The unfitted part of the signal is then taken as noise, so σ^2_{min} is probably an overestimation of the real amount of noise power present in the observations. As m

[4]Values m and σ^2_{p} are constant and known beforehand. The denominator, σ^2_0, is estimated via ML-II and is known to be close to $\frac{1}{n}\sum_{j=1}^{n} y_j^2 - \sigma^2$, as reasoned above.

[5]In this case only few hyperparameters need to be selected ($D + 2$ values in the case of the ARD kernel). This implies that only a small number of iterations is required for convergence, so that the overhead with respect to training a full MN is very small.

grows, more accurate estimations are expected, and in the concrete case of SSGP-fixed and MCN-fixed, we get closer to the corresponding stationary full GP.

Using an overestimated σ^2_{min} in BN-MN may force it to underfit. However, when m is small, BN-MN is expected to underfit anyway. And when m grows and the value of σ^2_{min} becomes more critical, so does the accuracy of σ^2_{min}.

Despite this rationale, when m is small, the amount of underfitting of MN-fixed is bigger than that of the corresponding MN (since the MN can adjust the input weights). This means that for small m, BN-MN is forced to underfit slightly more than MN. Thus, for small m, BN-MNs will perform slightly worse than MNs. This is the price paid in exchange for enhanced performance in a wide range of m values.

3.2.3 Bounded-Noise Marginalized Cosine Networks (BN-MCN)

In Section 2.6, before the concept of marginalized networks was formalized, we introduced MCNs as a relaxation of the phase integration property of SSGP. However, as the experiment in that section showed (Fig. 2.10), MCNs suffered of the general overfitting problem of MNs, whereas SSGP, due to the additional constraints placed on the input weights, did not overfit. Now we are in position to turn MCNs into a useful method by applying the noise bounding trick. We call the resulting algorithm Bounded-Noise Marginalized Cosine Networks (BN-MCN).

3.2.3.1 Model selection

As we have seen, BN-MNs have exactly the same structure as MNs, and use the same prediction (3.4) and NLML (3.4) equations, but they use a slightly different training procedure consisting of two steps: First a lower bound σ^2_{min} for the noise level is estimated keeping input weights fixed, then model selection is carried out while keeping noise lower-bounded.

This procedure is similar for any BN-MN; here, we provide details for BN-MCNs, since this is the method we will use in the experiments:

1. Run MCN-fixed:
   - Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, and σ^2 to some sensible values. (We will use: One half of the ranges of the input dimensions, the variance of the outputs $\{y_j\}_{j=1}^n$ and $\sigma_0^2/4$, respectively).
   - Fix $\{\boldsymbol{\omega}\}_{i=1}^m$ to random values drawn from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_D)$ (to approximate the ARD SE covariance function). Initialize $\{\varphi_i\}_{i=1}^m$ from a uniform distribution in $[0, 2\pi)$.
   - Minimize (3.5), the NLML of the model, wrt to σ_0^2, σ^2, $\{\ell_d\}_{d=1}^D$, and $\{\varphi_i\}_{i=1}^m$ (keeping $\{\boldsymbol{\omega}\}_{i=1}^m$ fixed).
2. Run BN-MCN:
   - Initialize $\{\ell\}_{d=1}^D$, σ_0^2, σ^2, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$ to the values they had after MCN-fixed converged.
   - Set σ_{min}^2 to the value found for σ^2 after convergence of MCN-fixed. Initialize σ^2 slightly above σ_{min}^2. (We will use $1.5\sigma_{\text{min}}^2$).
   - Minimize (3.5), the NLML of the model, wrt to $\{\ell_d\}_{d=1}^D$, σ_0^2, σ^2, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$, with the constraint $\sigma^2 > \sigma_{\text{min}}^2$.

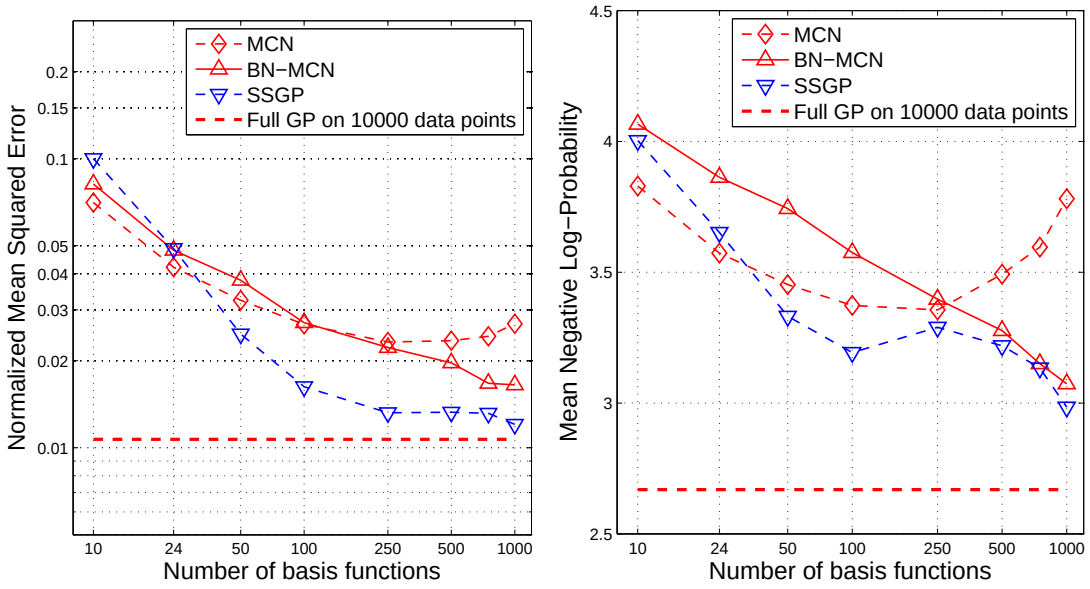
3.2.4 Experiments

In this section we will compare BN-MCNs with the method developed in the previous chapter, SSGP, as well as with the current state of the art for sparse regression, SPGP. Performance of a full GP on these tasks is also provided as a reference of state-of-the-art performance, only available at a high computational expense. Both SPGP and the full GP use the ARD SE covariance function, SSGP, MCN and BN-MCN are initialized to approximate it. For our code implementing SSGP, MCN and BN-MCN check Appendix F. For SPGP, the publicly available implementation from its authors is used.

All approximations run in $\mathcal{O}(m^2 n)$. To match the constant multiplicative factor, we will use the same number of basis functions for all methods (this matches the size of the involved matrices, so that computational cost becomes roughly identical). The number of spectral points/pseudo-inputs is therefore matched for BN-MCN/SPGP, whereas SSGP uses one spectral point per two pseudo-inputs (since it uses two basis functions per spectral point).

We will report as performance measures the test Normalized Mean Square Error (NMSE), which only checks the accuracy of the predictive means, and the test Mean Negative Log Probability (MNLP) which also takes into account the predictive variances. Both measures are described by (2.21). As usual, we report average values over ten repetitions.

3.2.4.1 The effect of noise bounding



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.1: NMSE and MNLP for MCN, BN-MCN, SSGP and full GP, for the *Pole Telecomm* problem.

In Section 2.6, we tried to use MCN as an alternative to SSGP on the *Pole Telecomm* data set. However, the result was unsatisfactory because some overfitting appeared. We can assess whether noise bounding is helpful to reduce it by looking at Fig. 3.1. As we knew, MCN overfits data starting at $m = 500$. We can see that BN-MCN does not. Predictive means and variances are also clearly improved for big m.

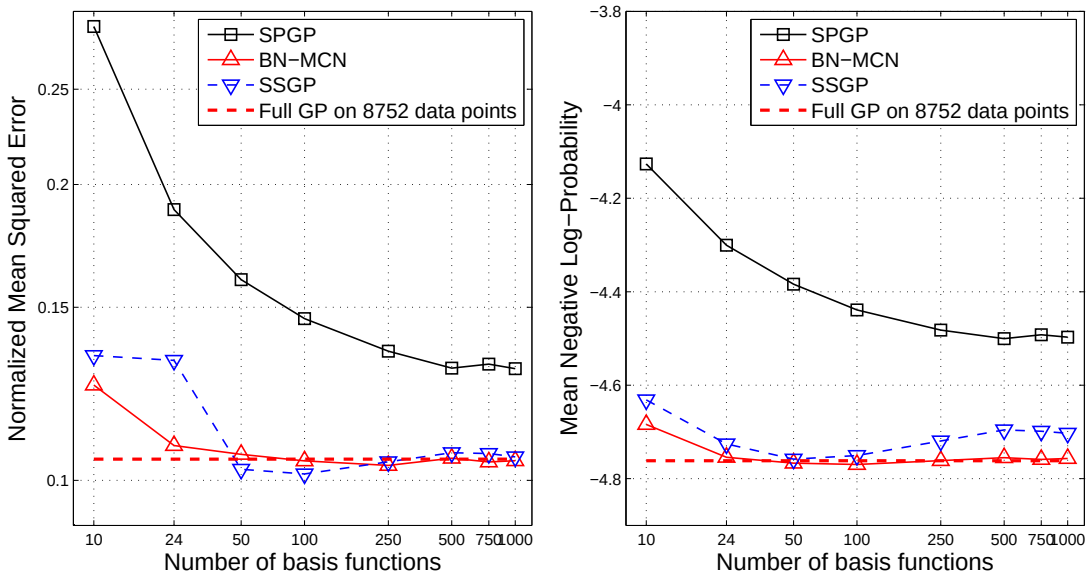
As described in Section 2.4.2, some unavoidable quantization noise $\text{bin}_{\text{spacing}}^2/12$ is present in this data set, so σ^2 is already lower bounded due to prior knowledge. According to the reasonings in Section 3.2.1, this lower bound provides MCN with some

overfitting protection. But BN-MCN has its own means to determine the appropriate lower bound and can raise[6] it as needed, which turns out to be helpful.

3.2.4.2 *Elevators* and *Pole Telecomm* pole data sets

Now we will compare the performances of SPGP, SSGP and BN-MCN on two large data sets, *Elevators* and *Pole Telecomm*, described in Section 2.4.2.

Results for *Elevators* are displayed in Fig. 3.2. BN-MCN performs slightly better than SSGP both in NMSE and NMLP terms, whereas both hugely outperform SPGP. No overfitting appears in the plots, despite using up to 1000 basis functions.



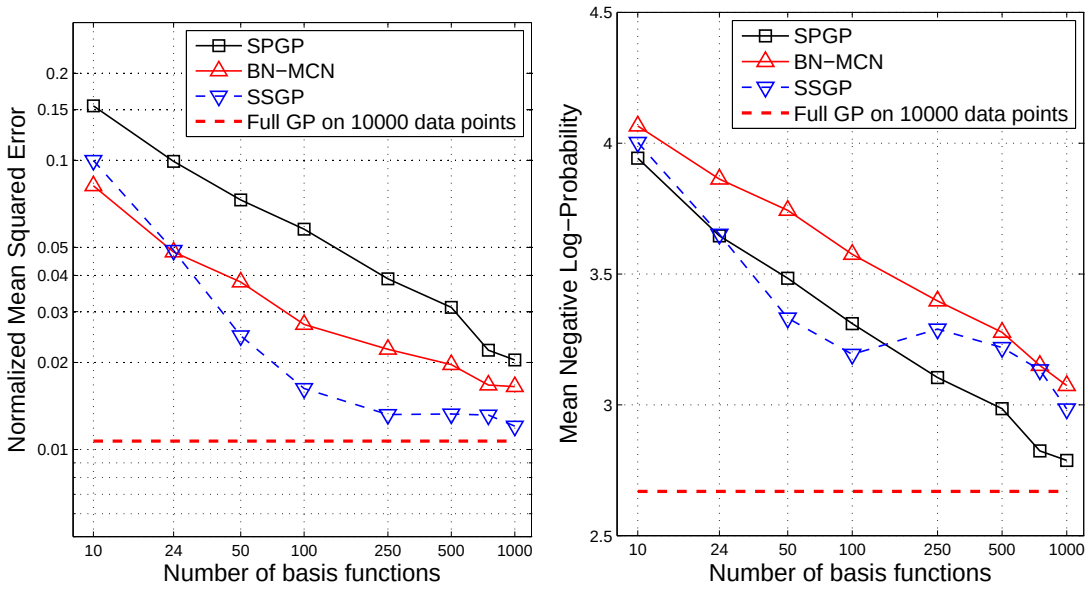
(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.2: NMSE and MNLP for SPGP, BN-MCN, SSGP and full GP, for the *Elevators* problem.

We have already seen the performance on *Pole Telecomm* of BN-MCN and SSGP compared with plain MCN in Fig. 3.1. In Fig. 3.3, we compare with our benchmark SPGP. BN-MCN manages to beat SPGP in NMSE but not on MNLP. No overfitting is reflected in any of these plots.

[6] While determining σ^2_{min} using MCN-fixed, the lower bound $\text{bin}^2_{\text{spacing}}/12$ is applied, so that $\sigma^2_{\text{min}} > \text{bin}^2_{\text{spacing}}/12$.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.3: NMSE and MNLP for SPGP, BN-MCN, SSGP and full GP, for the *Pole Telecomm* problem.

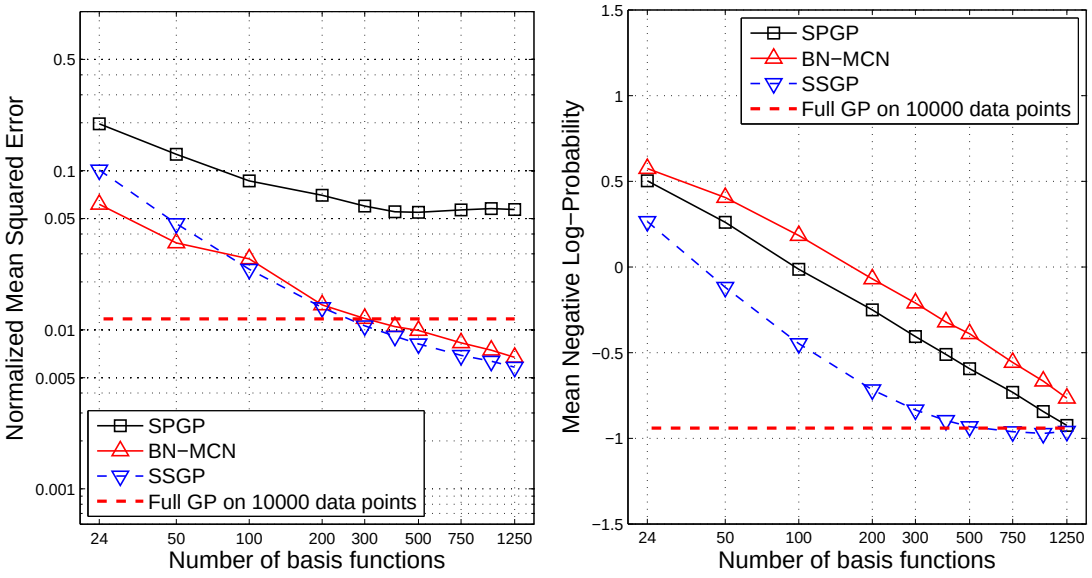
In conclusion, in these two data sets the predictive mean of BN-MCN is clearly superior to that of SPGP, as the NMSE plots show. The same cannot be said about the predictive variance, though it is not outrageous either.

3.2.4.3 *Kin-40k* and *Pumadyn-32nm* data sets

Now we consider the regression tasks presented in Seeger et al. (2003) and Snelson and Ghahramani (2006), as described in Section 2.4.2.

Fig. 3.4 displays results for the *Kin-40k* problem. As before, BN-MCN shows much better predictive means (smaller NMSE), whereas predictive variances are not so good (MNLP is slightly worse than SPGP).

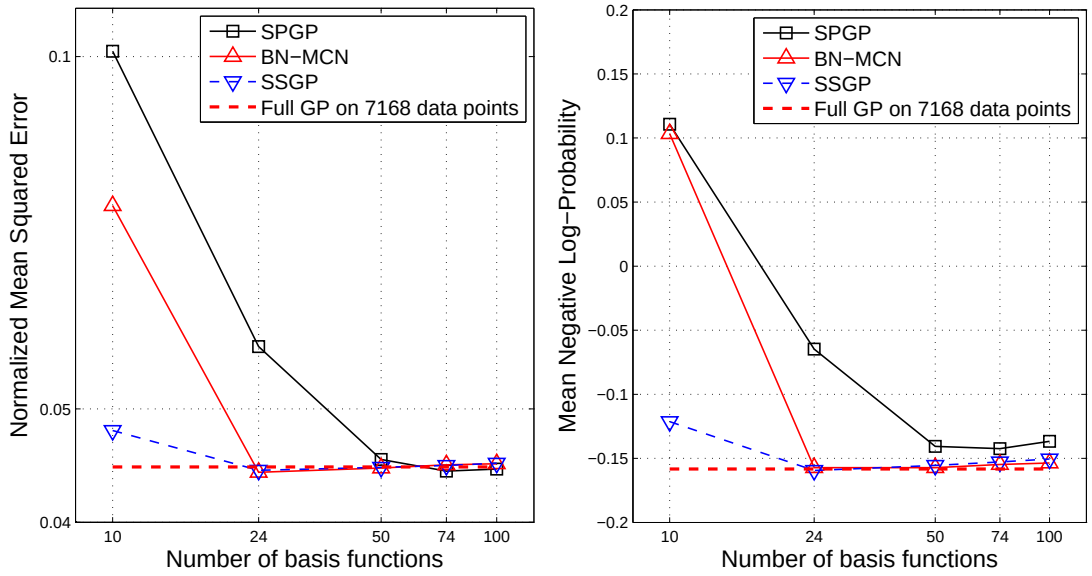
As we know, for problem *Pumadyn-32nm* only 4 out of the 32 input dimensions are useful to make predictions. Using the same length-scale initialization as the other methods, BN-MCN correctly performs ARD and singles them out to be dimensions $[4,\ 5,\ 15,\ 16]$. Its NMSE and MNLP are also better than those of SPGP.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.4: NMSE and MNLP for SPGP, BN-MCN, SSGP and full GP, for the *Kin-40k* problem.



(a) NMSE (log-log plot)

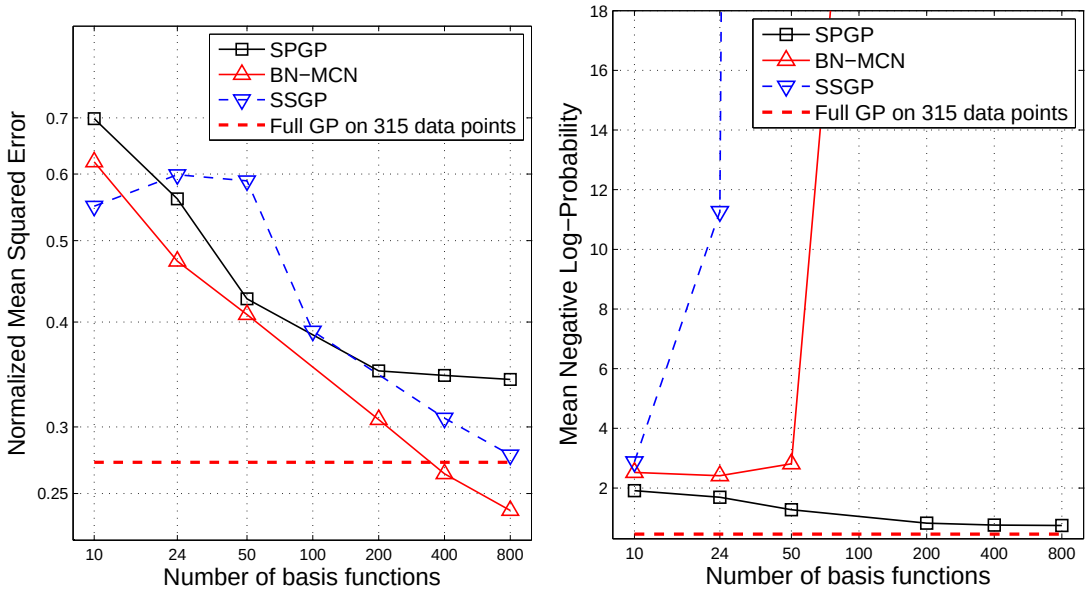
(b) MNLP (semilog plot)

Figure 3.5: NMSE and MNLP for SPGP, BN-MCN, SSGP and full GP, for the *Pumadyn-32nm* problem.

3.2.4.4 *Pendulum* data set

Now we turn to the "problematic" *Pendulum* data set, described in Section 2.4.4. Despite providing good predictive means, SSGP produced exceedingly bad predictive variances in this data set, for any m bigger than 10. Closer inspection revealed that for most test samples, the predicted variances were too small.

In Fig. 3.6, we see that this is the case for BN-MCN too (as expected, since BN-MCN does not provide particularly good predictive variances). Like SSGP, BN-MCN correctly determines the noise power present in data, so the small predictive variances are not due to noise underestimation.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.6: NMSE and MNLP for SPGP, BN-MCN, SSGP and full GP, for the *Pendulum* problem.

BN-MCN predictive means, as usual, are better than those of SPGP. It is interesting to notice how, using 800 basis, the state-of-the-art full GP is outperformed. Since we have only 315 data, computationally-wise it is not sensible to use a sparse method with more than 315 basis, since a full GP would be cheaper to train. However, if we have enough computing power, using more basis functions than data points can provide a performance boost on some problems.

3.2.4.5 Discussion

From the results obtained in all these data sets, it is possible to conclude that BN-MCNs (and more generally, BN-MNs) are a good replacement for classical networks:

(a) BN-MNs are much cheaper to train than a GP (the cost is similar to that of training a classical network, if Levenberg-Marquardt or a similar procedure is used).

(b) Predictive means (point estimates) with almost full-GP quality are provided.

(c) This quality does not degrade as m grows.

Furthermore, the presented approach can be considered as a novel way to train a classical network. Equation (3.4b) has network structure, so if we drop the probabilistic interpretation after training, we can consider the predictive means as the results obtained from a classical network.

Similar considerations can be made about SPGP, but two differences that favor BN-MNs are:

(a) The basis functions in SPGP must have the same form as the covariance function, whereas BN-MNs can be used with any family of basis functions.

(b) The mean performance of BN-MNs is better than that of SPGP.

SPGP provides more reliable predictive variances, but since predictive variances are not computed by classical networks anyway, this is not a disadvantage for tasks where the latter are used.

3.3 Marginalized Network Mixtures (MNmix)

MNs are defined (3.2) by integrating out the output weights of a general network structure (3.1). The input weights, however, have to be selected through ML-II so as to maximize the evidence. This design enforces sparsity, but results in two problems when m is big, as detailed in Section 3.1.3: Selecting many hyperparameters may result in overfitting, deteriorating the quality of both predictive means and variances; and the lack

of richness of the hypothesis space can produce overconfident predictive variances. Additionally, input weights selection may get stuck at inadequate local maxima of the evidence, but this is an unavoidable potential problem when working with sparse models[7]. Bayesian methods for NNs have been developed by Neal (1992, 1993, 1996), but they involve the use of Markov Chain Monte Carlo (MCMC) or other highly expensive methods.

A simple workaround against overfitting is to use noise bounding, as described in the previous section. This enhances both predictive means and variances, but the latter are still too unreliable.

In this section we propose an alternative solution based on combining several MNs that produces even better predictive means than noise bounding while almost removing the overconfidence problem, thus making them useful for cases where the full predictive posterior is required.

3.3.1 Combining MNs

Since the input weights of an MN are initialized randomly, a different model will be obtained for each run, corresponding to a different local optimum. With some approximations, it is possible to combine these different models into a single GP. Combining several different, independently trained models has a similar effect to "integrating out" the free parameters of the model. Since only a finite number of models are considered, this marginalization effect is only an approximation to proper Bayesian treatment.

We will approach the problem of combining several MNs from two different perspectives that lead to the same model: First, as a Gaussian approximation to the mixture model of the outputs of several MNs (which is simpler and computationally more efficient), and then, as a combination of the posteriors over the output weights (which more clearly elucidates the GP structure of the outputs under this model).

[7] In most cases, local optima found through (conjugate) gradient descent are good enough. The symmetry of the network, in which the roles of the basis functions can be freely interchanged and the sign of the input weights altered, implies that each (local or global) optimum can be obtained by a large number ($m!2^m$) of equivalent configurations.

3.3.1.1 MN mixture model and matching moments Gaussian

Consider first the full Bayesian approach in which the free parameters of the model are exactly integrated out. The predictive distribution of y_* at a new test point $\mathbf{x}_*$ is

$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p_{\text{MN}}(y_*|\mathbf{x}_*, \mathcal{M}, \mathcal{D}) p(\mathcal{M}|\mathcal{D}) \text{d}\mathcal{M}. \tag{3.6}$$

where $\mathcal{M} \equiv \{\sigma_0^2, \sigma^2, \mathbf{u}_1, \dots \mathbf{u}_m\}$ collects the signal and noise power hyperparameters and the input weights, i.e., the free parameters of the model.

Predictions using (3.6) will incur in no overfitting, since all parameters are integrated out. However, this integral is analytically intractable, so we will have to resort to numerical methods to compute it. Using Monte Carlo integration we have

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathcal{D}) \approx p_{\text{Mix}}(y_*|\mathbf{x}_*, \mathcal{D}) &= \frac{1}{K} \sum_{k=1}^{K} p_{\text{MN}}(y_*|\mathbf{x}_*, \mathcal{M}_k, \mathcal{D}) \\ p_{\text{MN}}(y_*|\mathbf{x}_*, \mathcal{M}_k, \mathcal{D}) &= \mathcal{N}(y_*|\mu_{\text{MN}*k}, \sigma^2_{\text{MN}*k}) \ , \end{aligned} \tag{3.7}$$

where $\{\mathcal{M}_k\}_{k=1}^K$ are samples drawn from $p(\mathcal{M}|\mathcal{D})$ and $\{\mu_{\text{MN}*k}, \sigma^2_{\text{MN}*k}\}$ are computed using (3.4) on the MN model described by $\mathcal{M}_k$. The Monte Carlo approximation converges to exact value in the infinite limit. Therefore, for $K = \infty$, (3.7) would coincide with the exact Bayesian posterior (3.6) and no overfitting would be present. When only a few samples are used (small K), the approximation is more coarse, but nonetheless introduces some overfitting resistance.

$\{\mathcal{M}_k\}_{k=1}^K$ must be sampled from the posterior probability $p(\mathcal{M}|\mathcal{D})$. This is not a trivial distribution and drawing samples from it usually requires to resort to expensive Markov Chain Monte Carlo Methods (MCMC). Since we only need a few samples (as little as 4 in our experiments), it is possible to use a much simpler strategy, such as choosing the samples to be the local modes of $p_{\text{MN}}(\mathcal{M}|\mathcal{D})$, i.e., high-probability samples according to our exploration of the distribution. Since the modes of $p(\mathcal{M}|\mathcal{D})$ correspond to the modes of the log-likelihood $\log p_{\text{MN}}(\mathbf{y}|\{\mathbf{x}_j\}_{j=1}^n, \mathcal{M})$ for a flat prior over $\mathcal{M}$, it is possible to obtain $\{\mathcal{M}_k\}_{k=1}^K$ as a set of local maximum likelihood estimates, which is the approach followed here. With these considerations, $\{p_{\text{MN}}(y_*|\mathbf{x}_*, \mathcal{M}_k, \mathcal{D})\}_{k=1}^K$ correspond to the posterior distributions of K independently trained MNs, which we assume to have converged to different modes of the likelihood. The mean and variance of the exact Bayesian posterior (3.6) can be approximated using

(3.7), yielding

$$\mu_{\text{MNmix}*} = \frac{1}{K}\sum_{k=1}^{K}\mu_{\text{MN}*k} \tag{3.8a}$$

$$\sigma^2_{\text{MNmix}*} = \frac{1}{K}\sum_{k=1}^{K}\mu^2_{\text{MN}*k} + \sigma^2_{\text{MN}*k} - \mu^2_{\text{MNmix}*} \tag{3.8b}$$

where $\mu_{\text{MN}*k}$ and $\sigma^2_{\text{MN}*k}$ are the predictive mean and variance provided by the k-th MN, respectively.

Though using a different motivation, our proposal is akin to bagging —Breiman (1996)—, which is known to reduce overfitting. Here we introduce diversity by combining several local modes of the evidence, instead of different subsamples of training data. Equation (3.8) is also presented in the context of GP bagging in Chen and Ren (2009).

Predictive distribution $p_{\text{Mix}}(y_*|\mathbf{x}_*, \mathcal{D})$ is not Gaussian, but a Gaussian mixture. Instead of directly using it, it is preferable to define the predictive distribution of the Marginalized Networks Mixture (MNmix) as the Gaussian that matches the moments of (3.7), which are given by (3.8), i.e., $p_{\text{MNmix}}(y_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(y_*|\mu_{\text{MNmix}*}, \sigma^2_{\text{MNmix}*})$. Thus, $p_{\text{MNmix}}(y_*|\mathbf{x}_*, \mathcal{D})$ is the Gaussian distribution with minimum Kullback-Leibler divergence from $p_{\text{Mix}}(y_*|\mathbf{x}_*, \mathcal{D})$.

The mean value predictions of MNmix correspond again to the architecture of a traditional network, this time with Km basis functions (with the output weights being $1/K$ times the output weights of each individual network).

Since MNs are trained independently and combining them is trivial, it is straightforward to asses the computational complexity of MNmix: $\mathcal{O}(Km^2n)$ time for training and $\mathcal{O}(Km^2)$ time for each probabilistic prediction. It is therefore possible to trade the number of basis functions for the number of networks and vice versa while remaining within the same computational complexity. In particular, halving the number of basis functions and using four times as many networks keeps computation time roughly identical, since one dependence is linear and the other quadratic. In practice, it turns out that even using a very small number of networks we get very good generalization abilities (almost no overfitting even for big m). Thus, in terms of computational complexity we can directly compare MNmix to SPGP, using a combination of four networks, each with half as many basis functions as pseudo-inputs are used in SPGP.

Analogously, we can compare with SSGP using four networks, each with the same number of basis functions as spectral points are used in SSGP.

3.3.1.2 MNmix as a posterior GP

The derivation above combines the outputs of K networks in a computationally efficient manner. However, though the marginal distributions over each output are approximated by Gaussians, it is unclear whether they jointly form a GP, i.e., if the joint posterior of the outputs corresponding to any possible set of inputs is a multivariate Gaussian distribution. In this section, we will derive a posterior GP (i.e., a GP given the data set) whose marginal distributions correspond to (3.8).

The posterior over the output weights for the k-th MN is

$$\begin{aligned} p(\mathbf{w}_k|\mathcal{D}) &= \mathcal{N}(\mathbf{w}_k|\mathbf{b}_k, \mathbf{\Sigma}_k) \\ \mathbf{b}_k &= \mathbf{A}_k^{-1}\mathbf{\Phi}_k\mathbf{y} \\ \mathbf{\Sigma}_k &= \sigma_k^2\mathbf{A}_k^{-1} \end{aligned}$$

where $\mathbf{A}_k$, $\mathbf{\Phi}_k$, and σ_k^2 correspond to the specific parametrization found for the k-th MN after training.

Output weights from different MNs are therefore independent. We can introduce dependencies among them by combining the independent posteriors in a single joint posterior as follows:

$$\begin{aligned} p(\tilde{\mathbf{w}}|\mathcal{D}) \equiv &\mathcal{N}(\tilde{\mathbf{w}}|\mathbf{b}, \mathbf{\Sigma}) \\ \mathbf{b} = &[\mathbf{b}_1^\top,\ \mathbf{b}_2^\top, \ldots,\ \mathbf{b}_K^\top]^\top \\ \mathbf{\Sigma} = &\,\mathrm{bd}(\sigma_1^2\mathbf{A}_1^{-1} + \mathbf{b}_1\mathbf{b}_1^\top, \ldots, \\ &\sigma_K^2\mathbf{A}_K^{-1} + \mathbf{b}_K\mathbf{b}_K^\top) \cdot K - \mathbf{b}\mathbf{b}^\top \end{aligned}$$

where $\tilde{\mathbf{w}} = [\mathbf{w}_1^\top,\ \mathbf{w}_2^\top, \ldots,\ \mathbf{w}_K^\top]^\top$, and $\mathrm{bd}(\cdot)$ arranges the matrices given as argument in block-diagonal form.

This joint Gaussian posterior preserves the posterior means of the separate MNs and introduces dependencies among the weights belonging to different networks (since the form of $\mathbf{\Sigma}$ is not entirely block-diagonal). The way posteriors are combined in $\mathbf{\Sigma}$

might look strange, but it is necessary to obtain the desired posterior marginal distributions. The output of the network mixture at any point $\mathbf{x}_*$ is defined as

$$y_{*\text{MNmix}}(\mathbf{x}_*) = \frac{1}{K}\sum_{k=1}^{K}(\mathbf{w}_k^\top \boldsymbol{\phi}_k(\mathbf{x}_*) + \sqrt{K}\varepsilon_k(\mathbf{x}_*))$$

i.e., the average of the outputs of each MN plus a white noise process. It is, therefore, a GP:

$$\begin{aligned} y_{*\text{MNmix}}(\mathbf{x}_*) &\sim \mathcal{GP}(m_{\text{MNmix}}(\mathbf{x}_*), k_{\text{MNmix}}(\mathbf{x}_*, \mathbf{x}'_*)) \\ m_{\text{MNmix}}(\mathbf{x}_*) &= \frac{1}{K}\mathbf{b}^\top \tilde{\boldsymbol{\phi}}(\mathbf{x}_*) \\ k_{\text{MNmix}}(\mathbf{x}_*, \mathbf{x}'_*) &= \frac{1}{K^2}\left[\tilde{\boldsymbol{\phi}}(\mathbf{x}_*)^\top \boldsymbol{\Sigma}\tilde{\boldsymbol{\phi}}(\mathbf{x}'_*) + \delta_{\mathbf{x}_*\mathbf{x}'_*} K \sum_{k=1}^{K}\sigma_k^2\right] \end{aligned} \tag{3.9}$$

where $\tilde{\boldsymbol{\phi}}(\cdot) = [\boldsymbol{\phi}_1(\cdot)^\top,\ \boldsymbol{\phi}_2(\cdot)^\top, \ldots,\ \boldsymbol{\phi}_K(\cdot)^\top]^\top$ is a column vector containing the outputs of the basis functions of all networks. Remember that this GP is a *posterior* process, not the usual GP prior imposed on $f(\mathbf{x})$.

Note that the mean and variance of $y_{*\text{MNmix}}(\mathbf{x}_*)$ at any point $\mathbf{x}_*$ are exactly those given by (3.8). Equation (3.8) is computationally preferable, whereas (3.9) provides richer details about the interdependence of the outputs of the mixture at different locations.

It is interesting to notice the way this networks mixture works: If we mix K networks that produce identical predictions, then mixture's predictions are also identical to the predictions of any of the involved networks, without any uncertainty reduction. Obtaining this effect is straightforward with (3.8), but to achieve the same effect in the derivation of this section, somewhat bizarre scalings with K must be introduced.

3.3.2 Marginalized Cosine Network Mixtures (MCNmix)

For the experiments, we will combine MNs with cosine basis functions, i.e., MCNs. We call this combination MCNmix. Since each MCN within MCNmix is trained independently, there is no specific model selection procedure. Instead, the complete procedure to compute predictive distributions using MCNmix is:

1. Train K different MCNs (using random initialization), as per Section 3.1.2.1 (i.e., minimize (3.5) wrt to $\{\ell\}_{d=1}^{D}$, σ_0^2, σ^2, $\{\boldsymbol{\omega}_i\}_{i=1}^{m}$ and $\{\varphi_i\}_{i=1}^{m}$).

2. Use the K MCNs to obtain the K predictive means and variances at new test points.

3. Combine them using equation (3.8).

Since training a single MCN is a $\mathcal{O}(m^2 n)$ process, total training cost for MCNmix is $\mathcal{O}(Km^2 n)$. Analogously, predictive means can be computed in $\mathcal{O}(Km^2)$ time per test point. Required storage space is dominated by the design matrices and is therefore $\mathcal{O}(Kmn)$.

3.3.3 Experiments

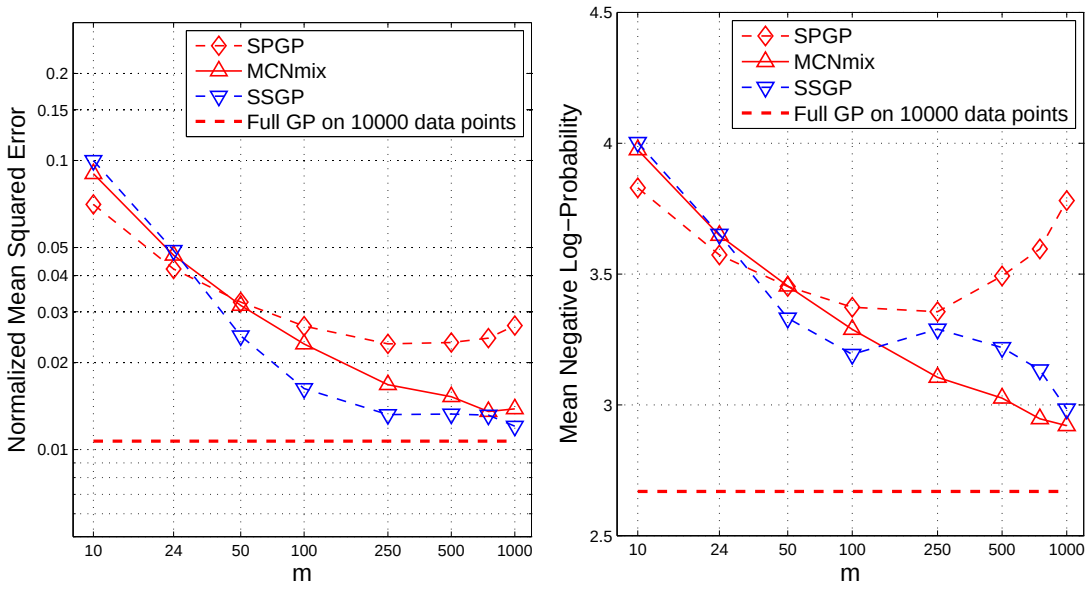
Now we will show how MCNmix improves over MCN and how it stacks up against SSGP and the current state of the art, SPGP, when run on our large data sets. Full GP performance is also quoted as a reference. Both SPGP and full GP use the ARD SE covariance function. SSGP, MCN and MCNmix are initialized to approximate it and then have their spectral points and input weights selected as described in Sections 2.3 3.1.2.1 and 3.3.2, respectively. For our code implementing SSGP, MCN and MCNmix check Appendix F. As before, for SPGP, the implementation from its authors is used.

All approximations run in $\mathcal{O}(m^2 n)$. Using the same number of basis functions m for SSGP, MCN and SPGP also matches their constant multiplicative factor so that they will all require roughly the same computation time. To match the multiplicative factor of MCNmix we will combine four networks ($K = 4$) with half the number of basis functions each ($m/2$), so that $\mathcal{O}(4(m/2)^2 n) = \mathcal{O}(m^2 n)$. Probabilistic predictions for new samples take $\mathcal{O}(m^2)$ per network, so the constant multiplicative factor is also matched at prediction time: $\mathcal{O}(4(m/2)^2) = \mathcal{O}(m^2)$. As noted before, the combination of a small number of networks is enough to achieve good results. Increasing this value would result in enhanced robustness at higher computational cost.

We will report NMSE and MNLP as performance measures, as described by (2.21). Recall that NMSE only measures the accuracy of the predictive means whereas MNLP also measures the accuracy of the predictive variances. As usual, we report average values over ten repetitions.

3.3.3.1 The effect of mixing

The expected benefit of using a mixture of MCNs (in general, MNs) appears in Fig. 3.7. Directly applying a MCN to the *Pole Telecomm* problem results, as we detailed in Section 2.6, in overfitting. However, when as little as four networks are mixed, results improve significantly. The improvement achieved by network mixing is better than that obtained through noise-bounding in Section 3.2.4.1, and the difference is bigger when the MNLP measure is considered. In general, network mixing is more effective than noise bounding, and provides more accurate predictive variances.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

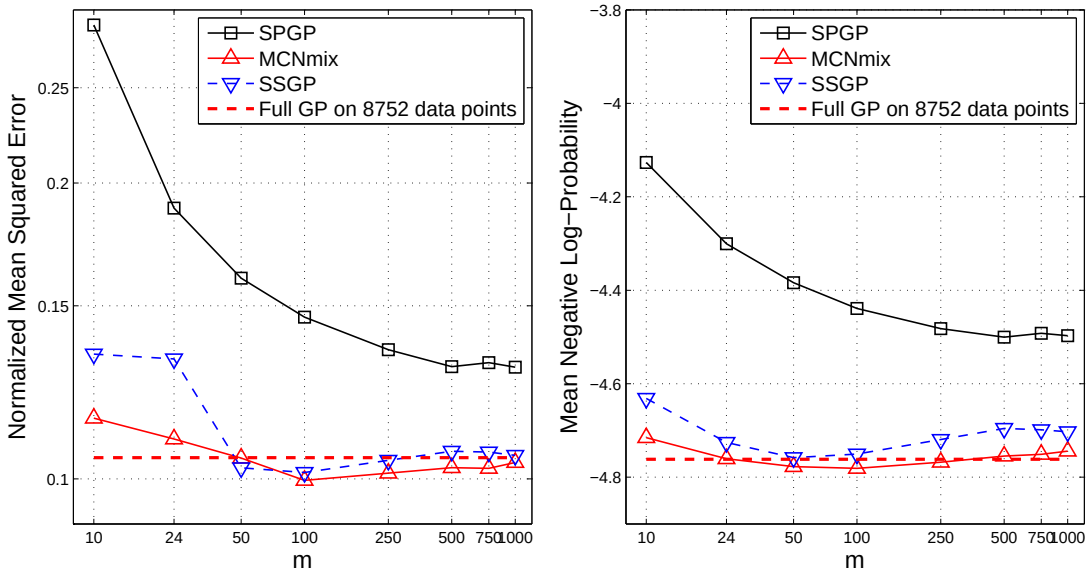
Figure 3.7: NMSE and MNLP for MCN, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Pole Telecomm* problem.

3.3.3.2 *Elevators* and *Pole Telecomm* pole data sets

Data sets *Elevators* and *Pole Telecomm* are described in Section 2.4.2.

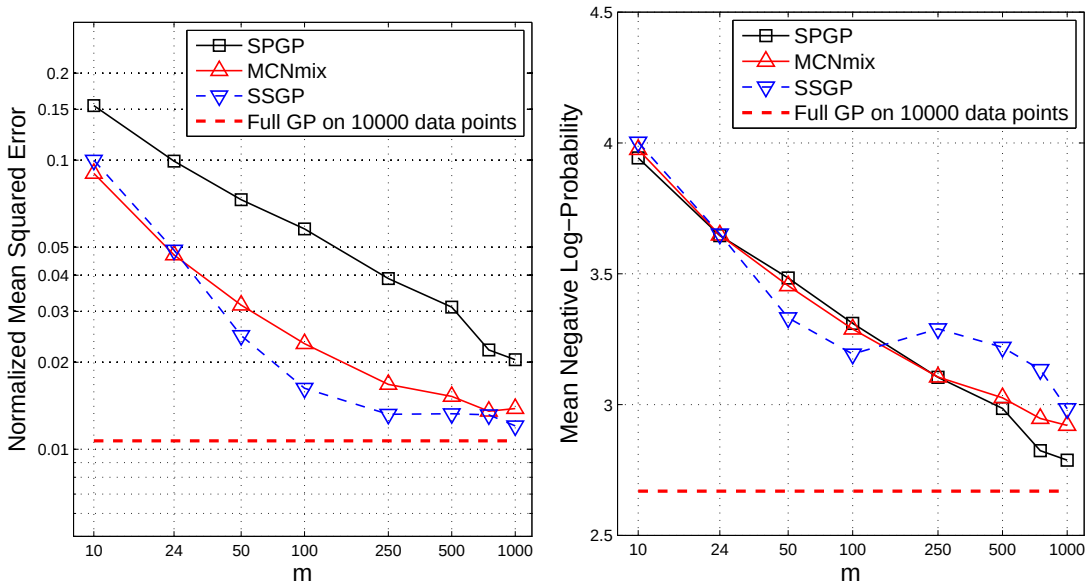
Fig. 3.8 shows results for the first data set. MCNmix provides slightly superior results than SSGP and a huge improvement with respect to SPGP, for both performance measures. The full GP is slightly outperformed, too.

For *Pole Telecomm*, Fig. 3.9, MCNmix and SSGP obtain clearly better results in NMSE, whereas similar results are obtained for the MNLP.



(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 3.8: NMSE and MNLP for SPGP, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Elevators* problem.



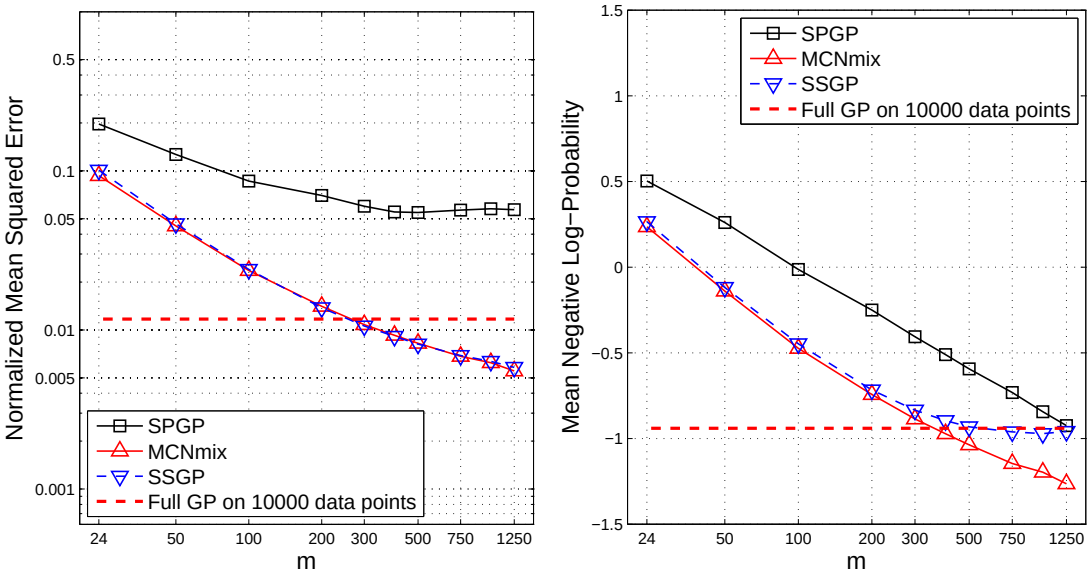
(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 3.9: NMSE and MNLP for SPGP, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Pole Telecomm* problem.

3.3.3.3 *Kin-40k* and *Pumadyn-32nm* data sets

See Section 2.4.3 for a description of these data sets.

Results for *Kin-40k* are reported in Fig. 3.10. MCNmix and SSGP perform similarly, with an advantage to MCNmix in MNLP for big m. The improvement in predictive accuracy achieved by the mixture of networks stands out: MCNmix outperforms the full GP both in terms of NMSE (as SSGP does) and MNLP (which SSGP is unable to do). SPGP is also clearly outperformed again.

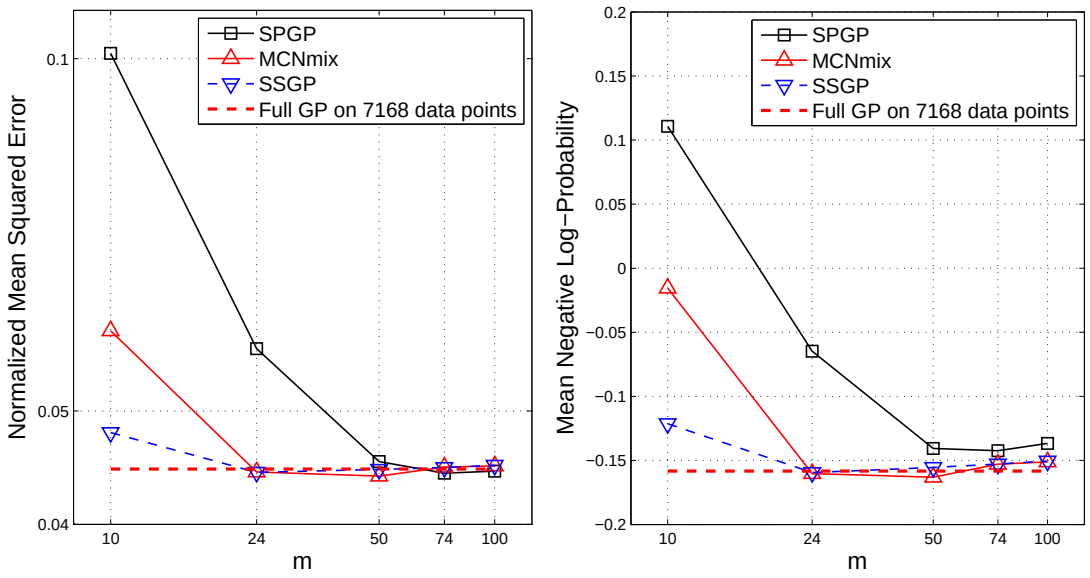


(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.10: NMSE and MNLP for SPGP, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Kin-40k* problem.

Fig. 3.11 shows the results for *Pumadyn-32nm*. Recall that only 4 out of the 32 input dimensions are useful to make predictions. Using the same length-scale initialization as the other methods, MCNmix correctly performs ARD and singles them out to be dimensions $[4,\ 5,\ 15,\ 16]$. Predictive means and variances keep being better than those of SPGP.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.11: NMSE and MNLP for SPGP, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Pumadyn-32nm* problem.

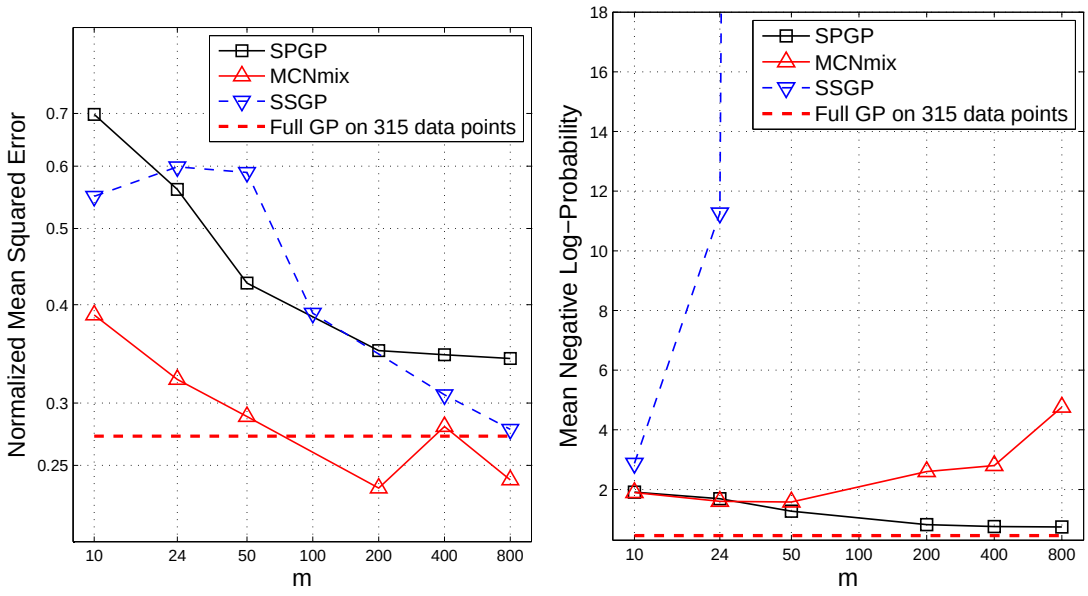
3.3.3.4 *Pendulum* data set

We consider once again the *Pendulum* data set, described in Section 2.4.4, for which both SSGP and BN-MCN produced exceedingly bad predictive variances.

In Fig. 3.12, we see that, although certainly predictive variances are not perfect, they are much more reasonable than those obtained by SSGP and BN-MCN: Up to $m = 50$, good predictive variances are obtained and up to $m = 400$ (which is even more than the number of data points, 315) the degradation of the MNLP is not drastic.

Even though the predictive variances obtained for this data set are not entirely satisfactory, they can be considered usable (it is also questionable whether anyone would want to take advantage of sparse methods with $m > 50$ in a data set with only 315 samples). If more robustness is required, it is always possible to increase the number of networks K.

On the other hand, MCNmix produces incredibly good NMSE results, much better than those of BN-MCN and also outperforming the full GP for large m.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 3.12: NMSE and MNLP for SPGP, SSGP (m basis functions), MCNmix (4 networks with $m/2$ basis functions) and full GP, for the *Pendulum* problem.

3.3.3.5 Discussion

The mixture of independent MNs seems a powerful approach. Mean predictions are clearly superior to those of SPGP, and on par or better than those of SSGP. Predictive variances are also reasonably good with respect to SPGP in most cases, and not as outrageous as those of SSGP or BN-MCN on problematic data sets.

This makes MCNmix a good replacement for classical networks (for the same reasons mentioned in Section 3.2.4.5) and a tough competitor among probabilistic sparse methods such as SPGP (though one should watch out for problematic data sets when the predictive variance is of interest, a problem that does not exist with SPGP).

It should also be noted that, if we are only interested in mean predictions, and we are going to train our model less often than we use it to make predictions, BN-MCN may be a better suited alternative: The cost of computing the predictive mean of a new test data point with MCNmix is $\mathcal{O}(4m/2) = \mathcal{O}(2m)$, whereas using BN-MCN is only $\mathcal{O}(m)$. Of course, this advantage disappears if predictive variances must also be computed.

3.4 Efficient supervised linear dimensionality reduction

In many types of MN, the basis functions can be expressed as a function of the dot product of the arguments

$$\phi([\varphi_i,\ \boldsymbol{\omega}_i^\top]^\top, \tilde{\mathbf{x}}_j) = \phi_{\mathrm{A}}(\boldsymbol{\omega}_i^\top \mathbf{x}_j + \varphi_i)$$

where $\phi_{\mathrm{A}}(\cdot)$ is a scalar function accepting a scalar argument. This function is usually called the "activation function". For this type of networks, the design matrix can be expressed as

$$\boldsymbol{\Phi}_{\mathbf{f}} = \phi_{\mathrm{A}}(\mathbf{W}\mathbf{X} + \boldsymbol{\varphi}\mathbf{1}_n^\top) \tag{3.10}$$

where $\mathbf{W} = [\boldsymbol{\omega}_1 \dots \boldsymbol{\omega}_m]^\top$, $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]$, $\boldsymbol{\varphi} = [\varphi_1 \dots \varphi_m]^\top$, $\mathbf{1}_n$ is a column vector with n ones and $\phi_{\mathrm{A}}(\cdot)$ is applied element-wise.

All models introduced so far, including SSGP, can be expressed in the form (3.10). This means that the only dependence of the design matrix on input data $\mathbf{X}$ is through the linear transformation $\mathbf{W}\mathbf{X}$. This fact can be used for supervised linear dimensionality reduction. After training a model, we can expand the obtained input weight matrix $\mathbf{W}$ using Singular Value Decomposition (SVD), as follows:

$$\mathbf{W} \overset{\text{SVD}}{\rightarrow} \mathbf{U}_{m\times D}\mathbf{S}_{D\times D}\mathbf{V}_{D\times D}^\top,$$

where the economy form of the SVD has been used. $\mathbf{S}$ is a matrix containing the singular values in non-increasing order. According to the Eckart-Young theorem, rank R matrix $\hat{\mathbf{W}}$ (with $R < D$) that best approximates $\mathbf{W}$ in the sense of minimizing the Frobenius norm of the difference is:

$$\mathbf{W} \approx \hat{\mathbf{W}} = \mathbf{U}_{m\times R}\mathbf{S}_{R\times R}\mathbf{V}_{D\times R}^\top,$$

where $\mathbf{U}_{m\times R}$ and $\mathbf{V}_{m\times R}$ result from truncating $\mathbf{U}_{m\times D}$ and $\mathbf{V}_{m\times D}$ to the first R columns and $\mathbf{S}_{R\times R}$ is a diagonal matrix with the first R singular values (the result of truncating $\mathbf{S}_{D\times D}$ to the first R rows and columns).

It is then possible to approximately re-express $\mathbf{W}\mathbf{X}$ as

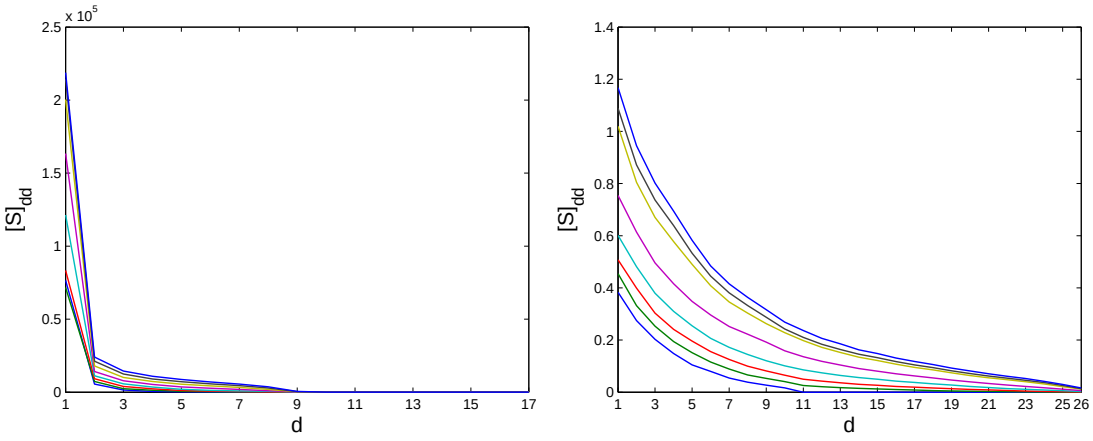
$$\mathbf{W}\mathbf{X} \approx \mathbf{U}_{m\times R}\hat{\mathbf{X}}_{R\times n} \ \text{ with } \ \hat{\mathbf{X}}_{R\times n} = \mathbf{Lp}_{R\times D}\mathbf{X} \ \text{ and } \ \mathbf{Lp}_{R\times D} = \mathbf{S}_{R\times R}\mathbf{V}_{D\times R}^\top$$

Matrix $\mathbf{Lp}_{R\times D}$ is a linear transformation that projects input data $\mathbf{X}$ to a space of dimension $R < D$. We can still use the trained model on the reduced-dimension data

$\hat{\mathbf{X}}_{R\times n}$ provided that we replace the original weight matrix $\mathbf{W}$ with a new weight matrix $\mathbf{U}_{m\times R}$. Note that there is no problem to make predictions at new data points, since we know how to transform them before applying the reduced-dimension model.

If we let R be the rank of $\mathbf{W}$ (which may be less than D), then the approximation becomes exact (since we are only truncating null singular values) and what we are really doing is finding the subspace where our model inherently lies. Only variations of the input within that subspace produce variations in the output.

The fraction $||\mathbf{S}_{R\times R}||_F/||\mathbf{S}_{D\times D}||_F$ (where $||\cdot||_F$ is the Frobenius norm) determines the amount of information that is retained after truncation, so we can let R be smaller than the rank of $\mathbf{W}$ as long as this fraction remains close to one. Since the projection matrix is optimal, for any given R the subspace containing the biggest amount of information from input data that is relevant for predictions will be selected.



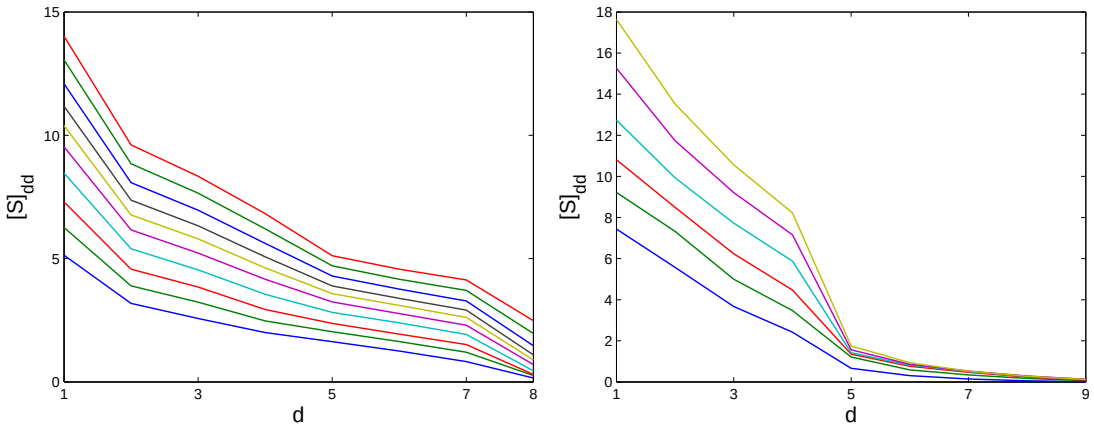
(a) *Elevators* (b) *Pole Telecomm*

Figure 3.13: Values of $[\mathbf{S}]_{dd}$ for the *Elevators* and *Pole Telecomm* data sets, when the number of basis functions is 10, 25, 50, 100, 250, 500, 750, 1000. Within each panel, plots portraying bigger singular values correspond to bigger values of m.

This dimensionality reduction method is also computationally attractive, since the only operation needed to compute the optimal projection matrix is an SVD on $\mathbf{W}$, which only takes $\mathcal{O}(mD^2)$, so it adds almost no overhead to the process of training a model.

We have computed the diagonal elements $\{[\mathbf{S}]_{dd}\}_{d=1}^{D}$ of matrix $\mathbf{S}_{D\times D}$ for the data sets considered in this chapter, using model BN-MCN and several values of m. Results

for *Elevators* and *Pole Telecomm* are displayed in Fig. 3.13 and results for *Kin-40k* and *Pendulum* are displayed in Fig. 3.14. Within each panel in the figures, several plots corresponding to different values of m are displayed, with biggest singular values corresponding to bigger values of m.

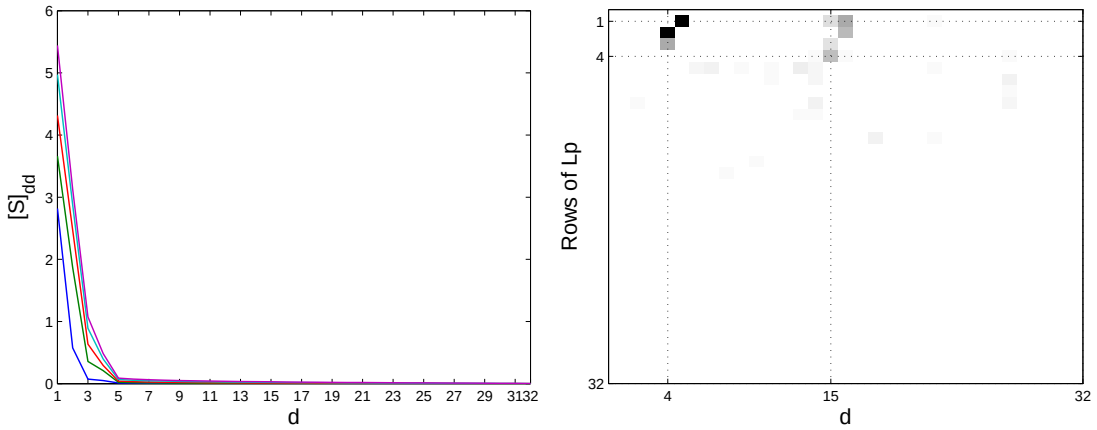


(a) *Kin-40k* (b) *Pendulum*

Figure 3.14: Values of $[\mathbf{S}]_{dd}$ for the *Kin-40k* and *Pendulum* data sets, when the number of basis functions are 25, 50, 100, 200, 300, 400, 500, 750, 1000, 1250 and 10, 25, 50, 200, 400, 800, respectively. Within each panel, plots portraying bigger singular values correspond to bigger values of m.

From these figures, it is possible to infer what is the effective dimension of these data sets for the purpose of regression. In *Pole Telecomm* and *Kin-40k*, there is no knee point that suggests linear dimensionality reduction can be performed. However, in *Elevators* it is possible to find a subspace of dimension 8 (out of 17 input dimensions) containing the same information as the original data. For *Pendulum* it seems that a subspace of dimension 4 (out of 9 input dimensions) would contain most of the relevant information, despite $\mathbf{S}_{D\times D}$ not being low rank.

For *Pumadyn-32nm* we knew that only 4 out of the 32 input dimensions were relevant. Fig. 3.15.(a) confirms this and Fig. 3.15.(b) shows the full projection matrix (if we set $R = 4$, only the first 4 rows are used, yielding a 4 dimensional projection). The linear projection correctly combines the relevant input dimensions 4, 5, 15 and 16.



(a) Singular values

(b) Projection matrix $\mathbf{Lp}$

Figure 3.15: (a) Values of $[\mathbf{S}]_{dd}$ for the *Pumadyn-32nm* data set, when the number of basis functions is 10, 25, 50, 75, 100. Plots portraying bigger singular values correspond to bigger values of m. (b) Full projection matrix $\mathbf{Lp}_{D\times D} = \mathbf{S}_{D\times D}\mathbf{V}_{D\times D}^{\top}$.

3.5 Summary and conclusions

In this chapter, instead of starting from an expensive full GP and trying to simplify it, we start from a traditional network and marginalize some, but not all, of their weights to get the key benefits of a GP without incurring in its high computational costs.

Using this idea, Marginalized Networks (MNs) arise as a probabilistic tool for regression that parallels classical frequentist regression networks. But MNs are not without problems, so we have proposed two alternative methods to improve them: Noise bounding and network mixing.

MNs with noise bounding have shown to be appropriate to replacement for old classical networks: While retaining the same structure and low computational cost, they can achieve close to full-GP NMSE performance. They can also be used to provide predictive variances, but often of poor quality.

Mixing several MNs provides better results both in terms of NMSE and MNLP than noise bounding, for the same computational cost. The quality of the predictions is in general similar or better than SSGP, and much better than SPGP. On some problematic data sets, predictive variances can be underestimated, but this problem is much

less significant than it is with SSGP or BN-MCN. MNmix can be considered as a replacement for classical networks, but its full probabilistic predictions are good enough to regard it as probabilistic sparse method.

Thus, we have provided two practical models for real world, large-scale regression, that stand in between classical networks and full Gaussian Processes, while trying to retain the advantages of both. Finally, a method for efficient dimensionality reduction has been suggested.

Chapter 4

Inter-Domain GPs

In this chapter we present a general inference framework for inter-domain GPs, and focus on its usefulness to build sparse GP models. The state-of-the-art SPGP model introduced by Snelson and Ghahramani (2006) relies on finding a small, representative pseudo data set of m elements (from the same domain as the n available data elements) which is able to explain those available data well, and then uses it to perform inference. This reduces inference and model selection computation time from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$, where $m \ll n$. Inter-domain GPs can be used to find a (possibly more compact) representative set of features lying in a different domain, at the same computational cost. Being able to specify a different domain for the representative features allows to incorporate prior knowledge about relevant characteristics of data and detaches the functional form of the covariance and basis functions. We will show how previously existing models, such as the one introduced by Walder et al. (2008), fit into this framework, and we will use it to develop two new sparse GP models. Tests on our already known large regression data sets suggest that significant improvement with respect to SPGP can be achieved, while retaining computational efficiency. This chapter is based on Lázaro-Gredilla and Figueiras-Vidal (2010).

The remainder of this chapter is organized as follows: In Section 4.1 we formalize the concept of Inter-Domain GPs (IDGPs); a sparse GP model framework based on IDGPs is presented in Section 4.2 and two concrete instantiations are derived in Section 4.3. Model selection is detailed in Section 4.4. Experimental results are provided in Section 4.5. Some general conclusions in Section 4.6 close the chapter.

4.1 Definition

In this section we will introduce Inter-Domain GPs (IDGPs), so that we can perform inference across different domains. This will allow to enhance SPGP by removing the constraint that the pseudo-inputs must remain within the same domain as input data. As we will see, this added flexibility results in increased performance and allows to encode prior knowledge about other domains where data can be represented more compactly.

Consider a real-valued GP $f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^D$ and some deterministic real function $g(\mathbf{x}, \mathbf{z})$, with $\mathbf{z} \in \mathbb{R}^H$. We define the following transformation:

$$u(\mathbf{z}) = \int_{\mathbb{R}^D} f(\mathbf{x}) g(\mathbf{x}, \mathbf{z}) d\mathbf{x} \ . \tag{4.1}$$

There are many examples of transformations that take on this form, the Fourier transform being one of the best known. We will discuss possible choices for $g(\mathbf{x}, \mathbf{z})$ in Section 4.3; for the moment we will deal with the general form. Since $u(\mathbf{z})$ is obtained by a linear transformation of GP $f(\mathbf{x})$, it is also a GP. This new GP may lie in a different domain of possibly different dimension. This transformation is not invertible in general, its properties being defined by $g(\mathbf{x}, \mathbf{z})$.

IDGPs arise when we jointly consider $f(\mathbf{x})$ and $u(\mathbf{z})$ as a single, "extended" GP. The mean and covariance function of this extended GP are overloaded to accept arguments from both the input and transformed domains and treat them accordingly. We refer to each version of an overloaded function as an *instance*, which will accept a different type of arguments. If the distribution of the original GP is $f(\mathbf{x}) \sim \mathcal{GP}(m_{\text{OR}}(\mathbf{x}),\ k_{\text{OR}}(\mathbf{x}, \mathbf{x}'))$, then it is possible to compute the remaining instances that define the distribution of the extended GP over both domains. The transformed-domain instance of the mean is

$$m_{\text{TR}}(\mathbf{z}) = \mathbb{E}[u(\mathbf{z})] = \int_{\mathbb{R}^D} \mathbb{E}[f(\mathbf{x})] g(\mathbf{x}, \mathbf{z}) d\mathbf{x} = \int_{\mathbb{R}^D} m_{\text{OR}}(\mathbf{x}) g(\mathbf{x}, \mathbf{z}) d\mathbf{x} \ .$$

The inter-domain instance of the covariance function is

$$\begin{aligned} k_{\text{INT}}(\mathbf{x}, \mathbf{z}') &= \mathbb{E}[f(\mathbf{x}) u(\mathbf{z}')] = \mathbb{E}\left[f(\mathbf{x}) \int_{\mathbb{R}^D} f(\mathbf{x}') g(\mathbf{x}', \mathbf{z}') \mathrm{d}\mathbf{x}' \right] \\ &= \int_{\mathbb{R}^D} k_{\text{OR}}(\mathbf{x}, \mathbf{x}') g(\mathbf{x}', \mathbf{z}') \mathrm{d}\mathbf{x}' \ , \end{aligned} \tag{4.2}$$

and the transformed-domain instance of the covariance function is

$$k_{\text{TR}}(\mathbf{z}, \mathbf{z}') = \mathbb{E}[u(\mathbf{z})u(\mathbf{z}')] = \mathbb{E}\left[\int_{\mathbb{R}^D} f(\mathbf{x})g(\mathbf{x}, \mathbf{z})d\mathbf{x} \int_{\mathbb{R}^D} f(\mathbf{x}')g(\mathbf{x}', \mathbf{z}')\mathrm{d}\mathbf{x}'\right] \tag{4.3}$$
$$= \int_{\mathbb{R}^D}\int_{\mathbb{R}^D} k_{\text{OR}}(\mathbf{x}, \mathbf{x}')g(\mathbf{x}, \mathbf{z})g(\mathbf{x}', \mathbf{z}')d\mathbf{x}\mathrm{d}\mathbf{x}' \ .$$

Mean $m(\cdot)$ and covariance function $k(\cdot, \cdot)$ are therefore defined both by the values and domains of their arguments. The latter select whether the appropriate instance for the covariance is $k_{\text{OR}}(\cdot, \cdot)$, $k_{\text{INT}}(\cdot, \cdot)$ or $k_{\text{TR}}(\cdot, \cdot)$ and whether $m_{\text{OR}}(\cdot)$ or $m_{\text{TR}}(\cdot)$ is used for the mean. This can be seen as if each argument had an additional domain indicator used to select the instance. Apart from that, they define a regular GP, and all standard properties hold. In particular $k(\mathbf{a}, \mathbf{b}) = k(\mathbf{b}, \mathbf{a})$.

This approach is related to Alvarez and Lawrence (2009), but here the latent space is defined as a transformation of the input space, and not the other way around. This allows to pre-specify the desired input-domain covariance. The transformation is also more general: Any $g(\mathbf{x}, \mathbf{z})$ can be used.

We can sample an IDGP at n input-domain points $\mathbf{f} = [f_1,\ f_2, \ldots,\ f_n]^\top$ (with $f_j = f(\mathbf{x}_j)$) and m transformed-domain points $\mathbf{u} = [u_1,\ u_2, \ldots,\ u_m]^\top$ (with $u_i = u(\mathbf{z}_i)$). With the usual assumption of $f(\mathbf{x})$ being a zero mean GP and defining $\mathbf{X} = \{\mathbf{x}_i\}_{j=1}^n$, $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^m$, the joint distribution of these samples is:

$$p\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{u}\end{bmatrix}\middle|\ \mathbf{X},\ \mathbf{Z}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{u}\end{bmatrix}\middle|\ \mathbf{0},\ \begin{bmatrix}\mathbf{K_{ff}} & \mathbf{K_{fu}}\\ \mathbf{K_{fu}}^\top & \mathbf{K_{uu}}\end{bmatrix}\right), \tag{4.4}$$
$$\text{with}\quad [\mathbf{K_{ff}}]_{pq} = k_{\text{OR}}(\mathbf{x}_p,\ \mathbf{x}_q), \quad [\mathbf{K_{fu}}]_{pq} = k_{\text{INT}}(\mathbf{x}_p,\ \mathbf{z}_q), \quad [\mathbf{K_{uu}}]_{pq} = k_{\text{TR}}(\mathbf{z}_p,\ \mathbf{z}_q),$$

which allows to perform inference across domains. We will only be concerned with one input domain and one transformed domain, but IDGPs can be defined for any number of domains.

4.2 Sparse regression using inducing features

In the standard regression setting, we are asked to perform inference about the latent function $f(\mathbf{x})$ from a data set $\mathcal{D}$ lying in the input domain. Using IDGPs, we can use data from any domain to perform inference in the input domain. Some latent functions might be better defined by a set of data lying in some transformed space rather than in the input space. This idea is used for sparse inference.

Following Snelson and Ghahramani (2006), we introduce a pseudo data set, but here we place it in the transformed domain: $\overline{\mathcal{D}} = \{\mathbf{Z}, \mathbf{u}\}$. The following derivation is analogous to that of SPGP. We will refer to $\mathbf{Z}$ as the *inducing features* and $\mathbf{u}$ as the inducing variables. The key approximation leading to sparsity is to set $m \ll n$ and assume that $f(\mathbf{x})$ is well-described by the pseudo data set $\overline{\mathcal{D}}$, so that any two samples (either from the training or test set) f_p and f_q with $p \neq q$ will be independent given $\mathbf{x}_p$, $\mathbf{x}_q$ and $\overline{\mathcal{D}}$. With this simplifying assumption, the prior over $\mathbf{f}$ can be factorized as a product of marginals:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Z}, \mathbf{u}) \approx \prod_{j=1}^{n} p(f_j|\mathbf{x}_j, \mathbf{Z}, \mathbf{u}). \tag{4.5}$$

Marginals are in turn obtained from (4.4): $p(f_j|\mathbf{x}_j, \mathbf{Z}, \mathbf{u}) = \mathcal{N}(f_j|\mathbf{k}_j\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \lambda_j)$, where $\mathbf{k}_j$ is the j-th row of $\mathbf{K}_{\mathbf{fu}}$ and λ_j is the j-th element of the diagonal of matrix $\mathbf{\Lambda}_{\mathbf{f}} = \mathrm{diag}(\mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}})$. Operator $\mathrm{diag}(\cdot)$ sets all off-diagonal elements to zero, so that $\mathbf{\Lambda}_{\mathbf{f}}$ is a diagonal matrix. Using this approximation, the joint prior on the latent and inducing variables is

$$p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z}) = p(\mathbf{f}|\mathbf{X}, \mathbf{Z}, \mathbf{u})p(\mathbf{u}|\mathbf{Z}) \approx p(\mathbf{u}|\mathbf{Z}) \prod_{j=1}^{n} p(f_j|\mathbf{x}_j, \mathbf{Z}, \mathbf{u}). \tag{4.6}$$

It is important to mention that the right hand side of (4.6) is the probability distribution that minimizes the Kullback-Leibler divergence from the exact joint prior (the left hand side of (4.6)) among all approximations of the form $p(\mathbf{u}|\mathbf{Z}) \prod_{j=1}^{n} q_j(f_j)$, for any possible $q_j(f_j)$, as argued in Snelson (2007, ch. 2). In other words, this is the best possible approximation of the full GP prior among those which assume that the latent variables are independent given the inducing variables (for a concrete set of inducing variables, which in turn is defined by the inducing features).

Since $p(\mathbf{u}|\mathbf{Z})$ is readily available and also Gaussian, the inducing variables can be integrated out from (4.5), yielding a new, approximate prior over $f(\mathbf{x})$:

$$\begin{aligned} p(\mathbf{f}|\mathbf{X}, \mathbf{Z}) &= \int p(\mathbf{f}|\mathbf{X}, \mathbf{Z}, \mathbf{u})p(\mathbf{u}|\mathbf{Z})d\mathbf{u} \approx \int \prod_{j=1}^{n} p(f_j|\mathbf{x}_j, \mathbf{Z}, \mathbf{u})p(\mathbf{u}|\mathbf{Z})d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}} + \mathbf{\Lambda}_{\mathbf{f}}). \end{aligned} \tag{4.7}$$

Using this approximate prior, the posterior distribution for a test case is:

$$p_{\text{IDGP}}(y_*|\mathbf{x}_*, \mathcal{D}, \mathbf{Z}) = \mathcal{N}(y_*|\mu_{\text{IDGP}*}, \sigma^2_{\text{IDGP}*}) \tag{4.8a}$$

$$\mu_{\text{IDGP}*} = \mathbf{k}_{\mathbf{u}*}^\top \mathbf{Q}^{-1} \mathbf{K}_{\mathbf{fu}}^\top \mathbf{\Lambda}_{\mathbf{y}}^{-1} \mathbf{y} \tag{4.8b}$$

$$\sigma^2_{\text{IDGP}*} = \sigma^2 + k_{**} + \mathbf{k}_{\mathbf{u}*}^\top (\mathbf{Q}^{-1} - \mathbf{K}_{\mathbf{uu}}^{-1}) \mathbf{k}_{\mathbf{u}*} \,, \tag{4.8c}$$

where we have defined $\mathbf{Q} = \mathbf{K}_{\mathbf{uu}} + \mathbf{K}_{\mathbf{fu}}^\top \mathbf{\Lambda}_{\mathbf{y}}^{-1} \mathbf{K}_{\mathbf{fu}}$ and $\mathbf{\Lambda}_{\mathbf{y}} = \mathbf{\Lambda}_{\mathbf{f}} + \sigma^2 \mathbf{I}_n$. Thus, the posterior distribution over test points (1.15) is approximated by (4.8) with the information available in the pseudo data set. After $\mathcal{O}(m^2 n)$ time precomputations, predictive means and variances can be computed in $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ time per test case, respectively. This model is, in general, non-stationary, even when it is approximating a stationary input-domain covariance, and can be interpreted as a degenerate GP plus heteroscedastic white noise.

The negative log-marginal likelihood of the model, explicitly including the conditioning on the kernel hyperparameters $\boldsymbol{\theta}$, can be expressed as

$$\begin{aligned} -\log p_{\text{IDGP}}(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) = +\frac{1}{2}[&\mathbf{y}^\top \mathbf{\Lambda}_{\mathbf{y}}^{-1} \mathbf{y} - \mathbf{y}^\top \mathbf{\Lambda}_{\mathbf{y}}^{-1} \mathbf{K}_{\mathbf{fu}} \mathbf{Q}^{-1} \mathbf{K}_{\mathbf{fu}}^\top \mathbf{\Lambda}_{\mathbf{y}}^{-1} \mathbf{y} \\ &+ \log(|\mathbf{Q}||\mathbf{\Lambda}_{\mathbf{y}}|/|\mathbf{K}_{\mathbf{uu}}|) + n \log(2\pi)] \end{aligned} \tag{4.9}$$

which is also computable in $\mathcal{O}(m^2 n)$ time.

Further computational savings and numerical accuracy can be achieved by using the Cholesky decomposition, as detailed in Section D.4 of Appendix D. Storage space is dominated by matrix $\mathbf{K}_{\mathbf{fu}}$, so that only $\mathcal{O}(nm)$ space is needed (as opposed to the $\mathcal{O}(n^2)$ space needed in a full GP to store the whole covariance matrix $\mathbf{K}_{\mathbf{ff}}$).

Model selection will be performed by jointly optimizing the evidence with respect to the hyperparameters and the inducing features. If analytical derivatives of the covariance function are available, conjugate gradient optimization also takes $\mathcal{O}(m^2 n)$ time per step, thus not increasing the complexity order of the complete training procedure.

4.3 On the choice of $g(\mathbf{x}, \mathbf{z})$

Feature extraction function $g(\mathbf{x}, \mathbf{z})$ defines the transformed domain in which the pseudo data set lies. According to (4.1), the inducing variables can be seen as projections of the target function $f(\mathbf{x})$ on the feature extraction function over the whole input space.

Therefore, each of them summarizes information about the behavior of $f(\mathbf{x})$ everywhere. The inducing features $\mathbf{Z}$ define the concrete set of functions over which the target function will be projected. It is desirable that this set captures the most significative characteristics of the function. This can be achieved either using prior knowledge about data to select $\{g(\mathbf{x}, \mathbf{z}_i)\}_{i=1}^m$ or using a very general family of functions and letting model selection automatically choose the appropriate set.

Another way to choose $g(\mathbf{x}, \mathbf{z})$ relies on the form of the posterior. The posterior mean of a GP is often thought of as a linear combination of "basis functions". For full GPs and other approximations – such as Csató and Opper (2002); Seeger et al. (2003); Smola and Bartlett (2001); Snelson and Ghahramani (2006); Tresp (2000); Williams and Seeger (2001) – basis functions must have the form of the input-domain covariance function. When using IDGPs, basis functions have the form of the inter-domain instance of the covariance function, and can therefore be adjusted by choosing $g(\mathbf{x}, \mathbf{z})$, independently of the input-domain covariance function.

If two feature extraction functions $g(\cdot, \cdot)$ and $h(\cdot, \cdot)$ can be related by $g(\mathbf{x}, \mathbf{z}) = h(\mathbf{x}, \mathbf{z})r(\mathbf{z})$ for any function $r(\cdot)$, then both yield the same sparse GP model. This property can be used to simplify the expressions of the instances of the covariance function.

In this work we use the same functional form for every feature, i.e. our function set is $\{g(\mathbf{x}, \mathbf{z}_i)\}_{i=1}^m$, but it is also possible to use sets with different functional forms for each inducing feature, i.e. $\{g_i(\mathbf{x}, \mathbf{z}_i)\}_{i=1}^m$ where each $\mathbf{z_i}$ may even have a different size.

In the subsections below we will discuss different possible choices for $g(\mathbf{x}, \mathbf{z})$, and will provide, for some input-domain covariance function, the corresponding inter-domain and transformed-domain instances of the covariance function.

4.3.1 Relation with Sparse GPs using pseudo-inputs (SPGP)

IDGP innovation with respect to SPGP consists in letting the pseudo data set lie in a different domain. If we set $g_{\text{SPGP}}(\mathbf{x}, \mathbf{z}) \equiv \delta(\mathbf{x} - \mathbf{z})$ where $\delta(\cdot)$ is a Dirac delta, we force the pseudo data set to lie in the input domain. Thus there is no longer a transformed space and the original SPGP model is retrieved. In this setting, the inducing features of IDGP play the role of SPGP's pseudo-inputs.

4.3.2 Relation with Sparse Multiscale GPs (SMGP)

Sparse Multiscale GPs (SMGPs) are presented in Walder et al. (2008). Seeking to generalize the SPGP model with ARD SE covariance function, they propose to use a different set of length-scales for each basis function. The resulting model presents a defective variance that is healed by adding heteroscedastic white noise. SMGPs, including the variance improvement, can be derived in a principled way as IDGPs:

$$g_{\text{SMGP}}(\mathbf{x}, \mathbf{z}) \equiv \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi(c_d^2 - \ell_d^2)}} \exp\left[-\sum_{d=1}^{D} \frac{(x_d - \mu_d)^2}{2(c_d^2 - \ell_d^2)}\right] \quad \text{with} \quad \mathbf{z} = \begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{c} \end{bmatrix} \tag{4.10a}$$

$$k_{\text{SMGP}}(\mathbf{x}, \mathbf{z}') = \exp\left[-\sum_{d=1}^{D} \frac{(x_d - \mu'_d)^2}{2c'^2_d}\right] \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{c'^2_d}} \tag{4.10b}$$

$$k_{\text{SMGP}}(\mathbf{z}, \mathbf{z}') = \exp\left[-\sum_{d=1}^{D} \frac{(\mu_d - \mu'_d)^2}{2(c_d^2 + c'^2_d - \ell_d^2)}\right] \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{c_d^2 + c'^2_d - \ell_d^2}}. \tag{4.10c}$$

With this approximation, each basis function has its own center $\boldsymbol{\mu} = [\mu_1,\ \mu_2, \ldots,\ \mu_d]^\top$ and its own length-scales $\mathbf{c} = [c_1,\ c_2, \ldots,\ c_d]^\top$, whereas global length-scales $\{\ell_d\}_{d=1}^D$ are shared by all inducing features. Equations (4.10b) and (4.10c) are derived from (4.2) and (4.3) using (1.5) and (4.10a). The integrals defining $k_{\text{SMGP}}(\cdot, \cdot)$ converge if and only if $c_d^2 \geq \ell_d^2, \forall_d$, which suggests that other values, even if permitted in Walder et al. (2008), should be avoided for the model to remain well defined.

4.3.3 Frequency Inducing Features GP (FIFGP)

If the target function can be described more compactly in the frequency domain than in the input domain, it can be advantageous to let the pseudo data set lie in the former domain. We will pursue that possibility for the case where the input domain covariance is the ARD SE. We will call the resulting sparse model Frequency Inducing Features GP (FIFGP).

Directly applying the Fourier transform is not possible because the target function is not square integrable (it has constant power σ_0^2 everywhere, so (4.3) does not converge). We will workaround this by windowing the target function in the region of interest. It is possible to use a square window, but this results in the covariance being defined in terms of the complex error function, which is very slow to evaluate. Instead,

we will use a Gaussian window[1]. Since multiplying by a Gaussian in the input domain is equivalent to convolving with a Gaussian in the frequency domain, we will be working with a blurred version of the frequency space. This model is defined by:

$$g_{\text{FIF}}(\mathbf{x}, \mathbf{z}) \equiv \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi c_d^2}} \exp\left[-\sum_{d=1}^{D} \frac{x_d^2}{2c_d^2}\right] \cos\left(\omega_0 + \sum_{d=1}^{D} x_d \omega_d\right) \text{ with } \mathbf{z} = \boldsymbol{\omega} \tag{4.11}$$

$$k_{\text{FIF}}(\mathbf{x}, \mathbf{z}') = \exp\left[-\sum_{d=1}^{D} \frac{x_d^2 + c_d^2 \omega_d'^2}{2(c_d^2 + \ell_d^2)}\right] \cos\left(\omega_0' + \sum_{d=1}^{D} \frac{c_d^2 \omega_d' x_d}{c_d^2 + \ell_d^2}\right) \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{c_d^2 + \ell_d^2}} \tag{4.12}$$

$$\begin{aligned} k_{\text{FIF}}(\mathbf{z}, \mathbf{z}') = \exp\left[-\sum_{d=1}^{D} \frac{c_d^2(\omega_d^2 + \omega_d'^2)}{2(2c_d^2 + \ell_d^2)}\right] \Bigg(\exp\left[-\sum_{d=1}^{D} \frac{c_d^4(\omega_d - \omega_d')^2}{2(2c_d^2 + \ell_d^2)}\right] \cos(\omega_0 - \omega_0') \\ + \exp\left[-\sum_{d=1}^{D} \frac{c_d^4(\omega_d + \omega_d')^2}{2(2c_d^2 + \ell_d^2)}\right] \cos(\omega_0 + \omega_0')\Bigg) \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{2c_d^2 + \ell_d^2}}. \end{aligned} \tag{4.13}$$

The inducing features are $\boldsymbol{\omega} = [\omega_0,\ \omega_1, \ldots,\ \omega_d]^\top$, where ω_0 is the phase and the remaining components are frequencies along each dimension. In this model, both global length-scales $\{\ell_d\}_{d=1}^D$ and window length-scales $\{c_d\}_{d=1}^D$ are shared, thus $c_d' = c_d$. Instances (4.12) and (4.13) are induced by (4.11) using (4.2) and (4.3).

Note that in this model the feature inducing function explicitly includes a phase, as MCN does. It is possible to integrate that phase out using twice as many inducing features, letting $\{\boldsymbol{\omega}_i\}_{i=1}^{2h} = \{[0\ \ \omega_1' \ldots \omega_D']_r^\top, [-\pi/2\ \ \omega_1' \ldots \omega_D']_r^\top\}_{r=1}^h$, as SSGP does.

4.3.4 Time-Frequency Inducing Features GP (TFIFGP)

Instead of using a single window to select the region of interest, it is possible to use a different window for each feature. We will use windows of the same size but different centers. The resulting model combines SPGP and FIFGP, so we will call it Time-Frequency Inducing Features GP (TFIFGP). It is defined by

[1] A mixture of m Gaussians could also be used as window without increasing the complexity order.

$$g_{\text{TFIF}}(\mathbf{x}, \boldsymbol{\omega}) \equiv \frac{1}{\prod_{d=1}^{D}\sqrt{2\pi c_d^2}} \exp\left[-\sum_{d=1}^{D}\frac{(x_d-\mu_d)^2}{2c_d^2}\right] \cos\left(\omega_0 + \sum_{d=1}^{D}(x_d-\mu_d)\omega_d\right) \text{ with } \mathbf{z} = \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\omega}\end{bmatrix} \tag{4.14}$$

$$k_{\text{TFIF}}(\mathbf{x}, \boldsymbol{\omega}') = \exp\left[-\sum_{d=1}^{D}\frac{(x_d-\mu'_d)^2 + c_d^2\omega_d'^2}{2(c_d^2+\ell_d^2)}\right] \cos\left(\omega'_0 + \sum_{d=1}^{D}\frac{c_d^2\omega'_d(x_d-\mu'_d)}{c_d^2+\ell_d^2}\right) \prod_{d=1}^{D}\sqrt{\frac{\ell_d^2}{c_d^2+\ell_d^2}} \tag{4.15}$$

$$\begin{aligned} k_{\text{TFIF}}(\boldsymbol{\omega}, \boldsymbol{\omega}') = \exp&\left[-\sum_{d=1}^{D}\frac{c_d^2(\omega_d^2+\omega_d'^2)+(\mu_d-\mu'_d)^2}{2(2c_d^2+\ell_d^2)}\right] \prod_{d=1}^{D}\sqrt{\frac{\ell_d^2}{2c_d^2+\ell_d^2}} \\ &\left(\exp\left[-\sum_{d=1}^{D}\frac{c_d^4(\omega_d-\omega'_d)^2}{2(2c_d^2+\ell_d^2)}\right]\cos(\omega_0-\omega'_0)\right. \\ &\left.+\exp\left[-\sum_{d=1}^{D}\frac{c_d^4(\omega_d+\omega'_d)^2}{2(2c_d^2+\ell_d^2)}\right]\cos(\omega_0+\omega'_0)\right) \end{aligned} \tag{4.16}$$

Note that this implies the following relations between FIFGP and TFIFGP:

$$\begin{aligned} g_{\text{TFIF}}(\mathbf{x}, \mathbf{z}) &= g_{\text{FIF}}(\mathbf{x}-\boldsymbol{\mu}, \boldsymbol{\omega}) \text{ with } \mathbf{z} = \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\omega}\end{bmatrix} \\ k_{\text{TFIF}}(\mathbf{x}, \mathbf{z}') &= k_{\text{FIF}}(\mathbf{x}-\boldsymbol{\mu}', \boldsymbol{\omega}') \\ k_{\text{TFIF}}(\mathbf{z}, \mathbf{z}') &= k_{\text{FIF}}(\mathbf{z}, \mathbf{z}')\exp\left[-\sum_{d=1}^{D}\frac{(\mu_d-\mu'_d)^2}{2(2c_d^2+\ell_d^2)}\right] \end{aligned}$$

FIFGP is trivially obtained by setting every center to zero, $\{\boldsymbol{\mu}_i = \mathbf{0}\}_{i=1}^{m}$, whereas SPGP is obtained by setting window length-scales $\mathbf{c}$ and frequencies and phases $\{\boldsymbol{\omega}_i\}_{i=1}^{m}$ to zero. If the window length-scales were individually adjusted, SMGP would be obtained.

While FIFGP has the modeling power of both FIFGP and SPGP, it might perform worse in practice due to it having roughly twice as many hyperparameters, thus making the optimization problem harder. The same problem also exists in SMGP. A possible workaround is to initialize the hyperparameters using a simpler model, as done in Walder et al. (2008) for SMGP, though we will not do this here.

The same strategy used with FIFGP to integrate phases out can be applied here, too.

4.4 Model selection

Model selection is performed as usual, minimizing the NLML of the model (ML-II). Note that in contrast with methods from previous chapters, no overparametrization is necessary for the length-scales to appear as separate hyperparameters. The detailed procedure to train FIFGP and TFIFGP follows:

1. Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, σ^2, $\{c_d\}_{d=1}^D$ to some sensible values. (We will use: One half of the ranges of the input dimensions, the variance of the outputs $\{y_j\}_{j=1}^n$, $\sigma_0^2/4$ and the standard deviation of input data, respectively). For TFIFGP, also let $\{\boldsymbol{\mu}_i\}_{i=1}^m = \mathbf{0}$.

2. Initialize $\{\boldsymbol{\omega}_i\}_{i=1}^m$ from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{L}^{-2})$ and $\{\omega_{0i}\}_{i=1}^m$ from a uniform distribution in $[0, 2\pi)$.

3. Minimize (4.9), the NLML of the model, wrt to $\{\ell_d\}_{d=1}^D$, σ_0^2, σ^2, $\{c_d\}_{d=1}^D$ and $\{\boldsymbol{\omega}_i\}_{i=1}^m$. For TFIFGP, also wrt $\{\boldsymbol{\mu}_i\}_{i=1}^m$. Since analytical derivatives are available, conjugate gradient descent can be used.

The initialization used in step 2 is taken from the initialization of MCN: If we used no windowing ($\{c_d\}_{d=1}^D = \infty$), both initializations would yield the same set of basis functions, so it is probably a reasonable starting point. Detailed and numerically robust equations to compute the derivatives of the NLML in step 3 are provided in Section E.3 of Appendix E.

4.5 Experiments

We will now test the ability of FIFGP and TFIFGP to perform sparse regression on our large-scale data sets and compare them with SPGP and a full GP. In all cases, the (input-domain) covariance function is the ARD SE (1.5). All methods run in $\mathcal{O}(m^2 n)$. To match computation time among the sparse algorithms, the same number of basis functions (i.e. inducing features/pseudo-inputs) is used in all methods. As usual, averages over ten repetitions of NMSE and MNLP figures are reported for each data set, for several choices of the number of basis functions. Unlike the other methods, FIFGP

is not translation invariant[2] (due to the Gaussian window of $g_{\text{FIF}}(\mathbf{x}, \mathbf{z})$ being specifically centered at zero), so we will additionally center all data sets (by substracting the sample mean from all inputs). This move does not affect the other algorithms.

Unlike the methods proposed in previous chapters, FIFGP and TFIFGP cannot be interpreted as generalized linear regressors. They are also more constrained methods because their approximate prior (4.6) is known to have minimum Kullback-Leibler distance from the true prior, as explained in Section 4.2. Intuitively, this constraint will force them to follow the full GP more closely (and will therefore render them less prone to overfitting or producing inadequate predictive variances), but will also avoid the flexibility that allowed previous models to have high performance in the sparser regime.

4.5.1 *Elevators* and *Pole Telecomm* data sets

We start by considering the large regression data sets *Elevators* (Fig. 4.1) and *Pole Telecomm* (Fig. 4.2), described in Section 2.4.2.

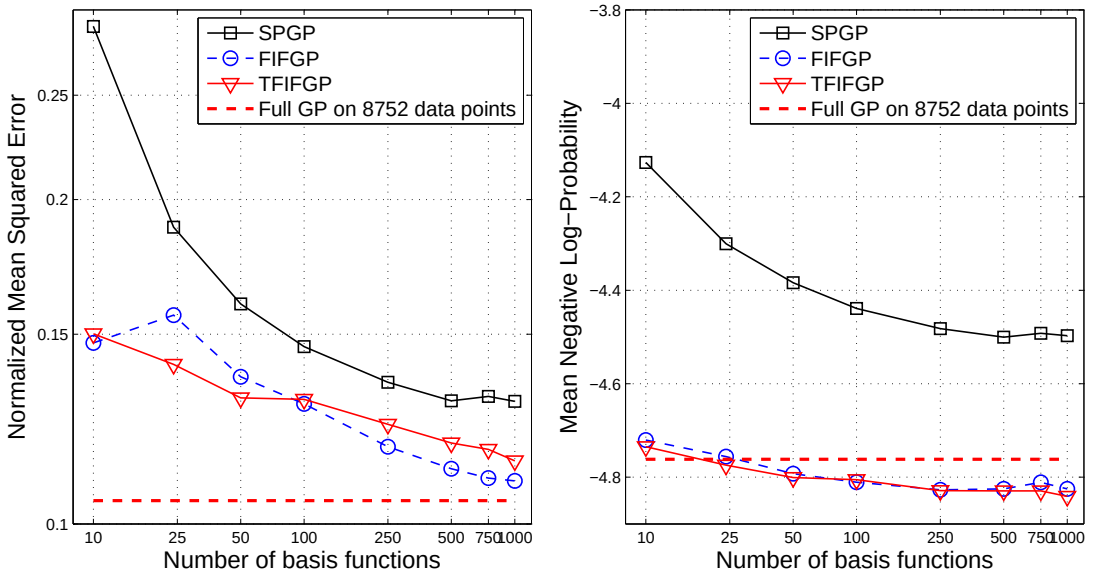
Both FIFGP and TFIFGP show a clear improvement over SPGP in terms of NMSE, for both data sets. In terms of MNLP, there is a huge improvement for *Elevators*, whereas there is little difference for *Pole Telecomm*. Quantization noise was taken into account for this second data set, as described in previous chapters. We have explicitly confirmed that if quantization noise was not considered, the MNLP performance of all methods would increase slightly.

4.5.2 *Kin-40k* and *Pumadyn-32nm* data sets

For *Kin-40k*, Fig. 4.3, all three sparse methods perform similarly, though for high sparseness (the most useful case) FIFGP and TFIFGP are slightly superior.

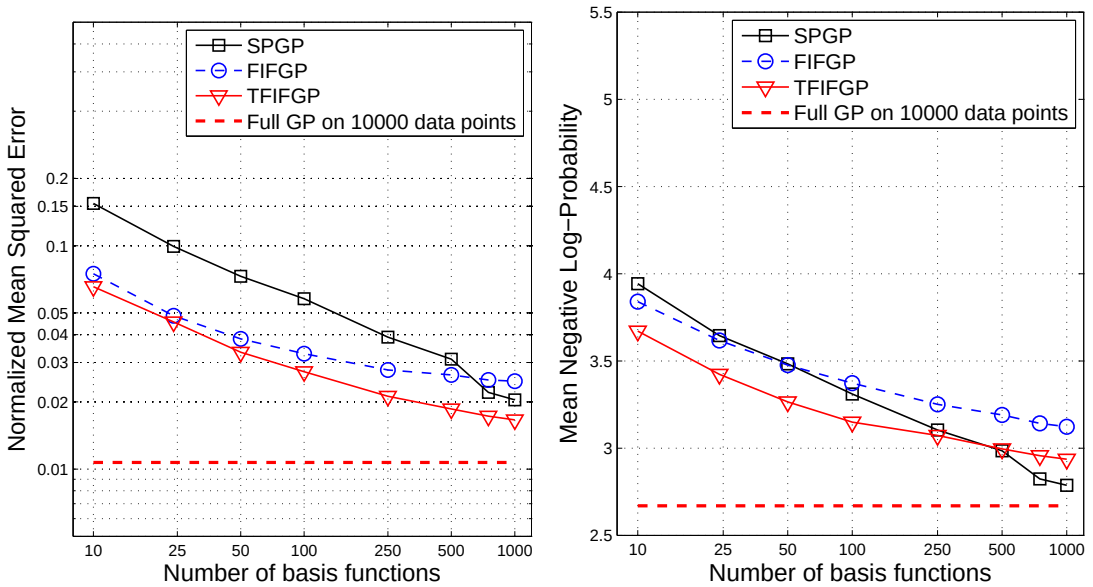
In *Pumadyn-32nm*, Fig. 4.4, we already know that only 4 out the 32 input dimensions are relevant to the regression task, so it can be used as an ARD capabilities test. We follow Snelson and Ghahramani (2006) and use a full GP on a small subset of the training data (1024 data points) to obtain the initial length-scales for all methods. This

[2]FIFGP can be easily made translation invariant by regarding its Gaussian window center as an additional hyperparameter.



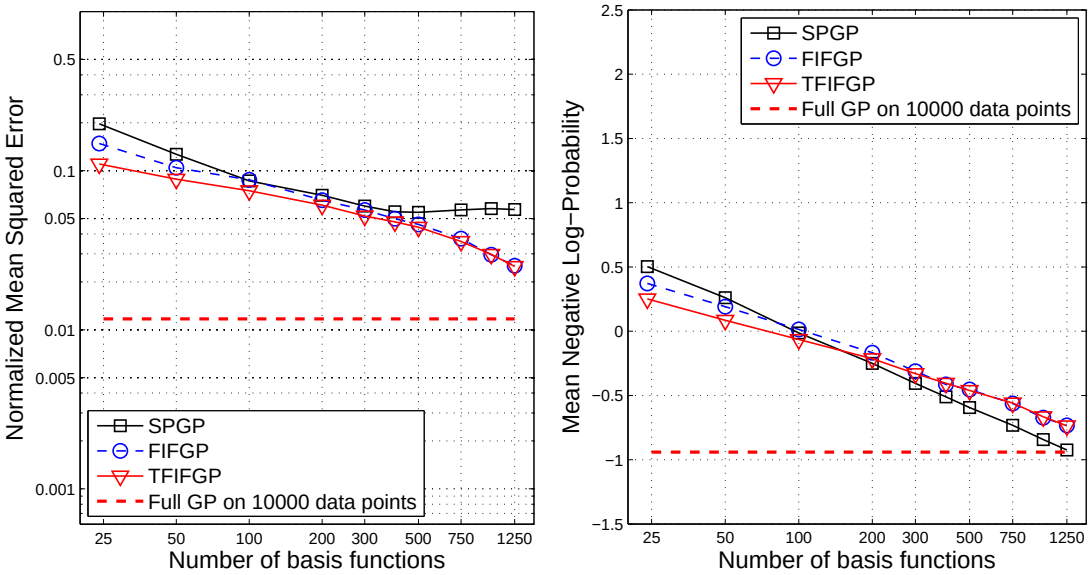
(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 4.1: NMSE and MNLP for SPGP, FIFGP, TFIFGP and full GP, for the *Elevators* problem.



(a) NMSE (log-log plot) (b) MNLP (semilog plot)

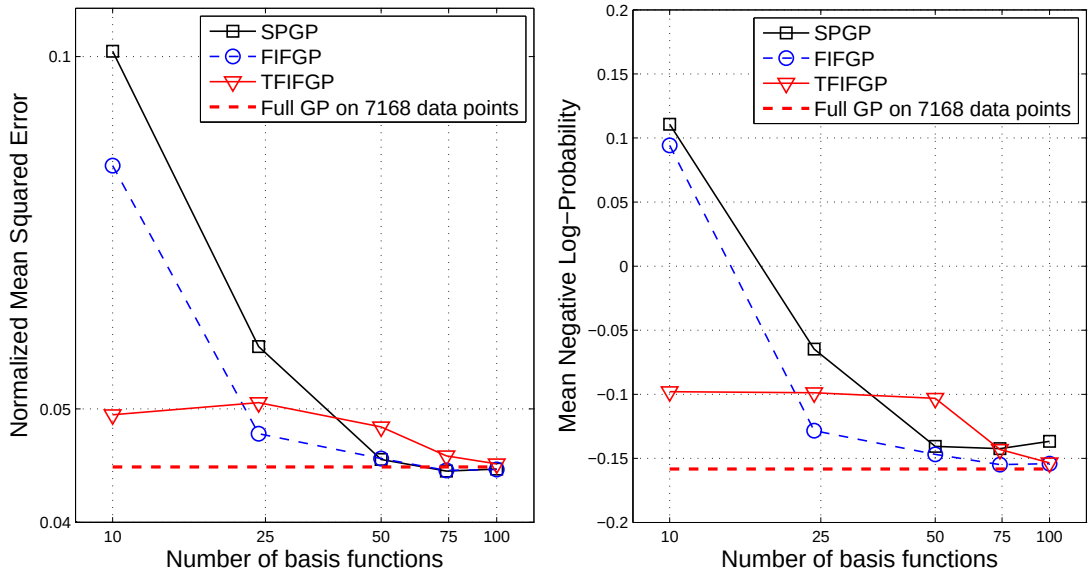
Figure 4.2: NMSE and MNLP for SPGP, FIFGP, TFIFGP and full GP, for the *Pole Telecomm* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 4.3: NMSE and MNLP for SPGP, FIFGP, TFIFGP and full GP, for the *Kin-40k* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

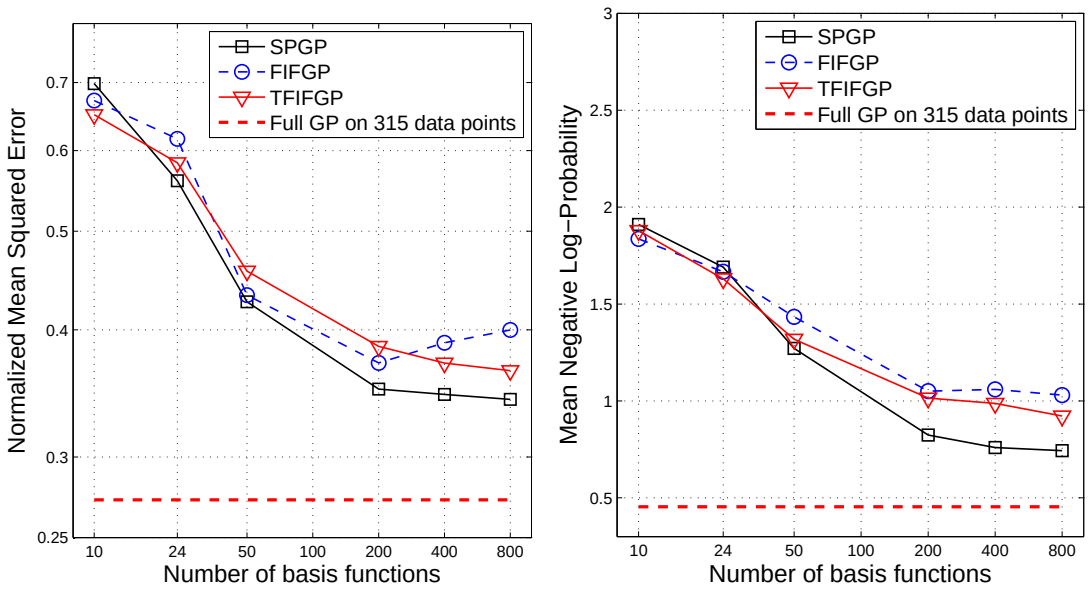
Figure 4.4: NMSE and MNLP for SPGP, FIFGP, TFIFGP and full GP, for the *Pumadyn-32nm* problem.

allows better minima to be found during optimization. All methods are able to properly find the correct solution: The relevant dimensions are $[4,\ 5,\ 15,\ 16]$. However, FIFGP and especially TFIFGP produce better results in the sparser regime.

4.5.3 *Pendulum* data set

It has been shown that for the *Pendulum* data set, described in Section 2.4.4, it was difficult to obtain accurate predictive variances. This problem appeared clearly when using SSGP and BN-MCN and, to a lesser extent when using MCNmix.

TFIFGP and FIFGP are approximations of a different nature, which properly handle the uncertainties derived from the sparsity of the model. Fig. 4.5 shows the behavior of these models with data set *Pendulum*.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 4.5: NMSE and MNLP for SPGP, FIFGP, TFIFGP and full GP, for the *Pendulum* problem.

Even using $m > n$ and despite the relatively high number of parameters being adjusted, proper predictive variances are obtained.

4.6 Summary and conclusions

In this chapter we have introduced IDGPs, which are able to combine representations of a GP in different domains, and have used them to extend SPGP to handle inducing features lying in a different domain. This provides a general framework for sparse models, which are defined by a feature extraction function. Using this framework, MSGPs proposed by Walder et al. (2008) can be reinterpreted as fully principled models using a transformed space of local features, without any need for post-hoc variance improvements. Furthermore, it is possible to develop new sparse models of practical use, such as the proposed FIFGP and TFIFGP, which are able to outperform the state-of-the-art SPGP on some large data sets, especially for high sparsity regimes.

Choosing a transformed space for the inducing features enables to use domains where the target function can be expressed more compactly, or where the evidence (which is a function of the features) is easier to optimize. This added flexibility translates as a detaching of the functional form of the input-domain covariance and the set of basis functions used to express the posterior mean.

IDGPs (including SPGP) approximate full GPs optimally in the KL sense noted in Section 4.2, *for a given set of inducing features*. Using ML-II to select the inducing features means that models providing a good fit to data are given preference over models that might approximate the full GP more closely. This, though rarely, might lead to harmful overfitting. To more faithfully approximate the full GP and avoid overfitting altogether, our proposal can be combined with the variational approach from Titsias (2009), in which the inducing features would be regarded as variational parameters. This would result in more constrained models, which would be closer to the full GP but might show reduced performance.

IDGPs as a general tool can be used for other purposes, such as modeling noise in the frequency domain, aggregating data from different domains or even imposing constraints on the target function.

Models arising from the IDGP framework (including SPGP) are more constrained than than the models seen in previous chapters (SSGP, BN-MN, MNmix), so the improvements of FIFGP and TFIFGP over SPGP are smaller. On the other hand, the constraints of IDGP models allow them to properly handle the uncertainties derived from the finiteness of the function set, so that no data set results in poorly determined

predictive variances. When considering an application in which the accuracy of every predictive variance is an absolute must, this type of models are recommended over those of previous chapters.

Chapter 5

Extensions

The ideas introduced in previous chapters are readily amenable to variations and extensions, thus yielding a plethora of potentially useful algorithms. Without intending to be exhaustive, in this chapter we show a few concrete extensions that we consider to be of special interest or not-so-obvious development. They are:

- Multi-Layer Perceptrons as MNs: Marginalized Networks (MNs) were introduced in Chapter 3, along with noise bounding and network mixing to render them useful. However, our experiments were limited to the case of cosine-type activations, so as to maintain the connection with SSGP. One might wonder if the usefulness of these improved MNs can be extended to other activation functions. In Section 5.1 we address this question by considering MNs with sigmoid activation function, which mimic the structure of Muti-Layer Perceptrons (MLPs). The performance of SPGP with an "MLP" covariance function, a possibility largely ignored so far in the literature, is also investigated.

- Non-Gaussian likelihoods: If any of the methods presented so far is coupled with a non-Gaussian likelihood, approximate inference and model selection are still possible within the same complexity order. Details are provided in Section 5.2.

- Sparse robust regression and classification: Varying the likelihood function, it is possible to perform robust regression and classification. This is illustrated in Section 5.3 using the BN-MCN model of Chapter 3 to perform robust regression and in Section 5.4 using the FIFGP model of Chapter 4 to perform classification. Experiments with these new models are also provided.

5.1 Muti-Layer Perceptrons (MLPs) as MNs

In this section we first review MLPs and how they converge to full GPs as the number of hidden units tends to infinity (provided that Gaussian priors are placed on the network weights). Then we marginalize MLPs and combine them with noise bounding and network mixing, as described in Chapter 3. Finally, we provide experimental results on our large data sets. We also include results for SPGP with MLP covariance function, since under certain conditions it converges to the same full GP as a marginalized MLP.

5.1.1 Multi-Layer Perceptrons

Neural Networks are a biologically inspired computational model which consists on a set of interconnected processing units or "neurons". They can be arranged in several different structures, but here we will only be concerned with feedforward networks with one hidden layer and sigmoid activation function, also known as MLPs. This architecture stands out because it has been proved that, for a sufficiently large number of hidden units, it can approximate a wide range of functions, with the notable exception of polynomials, see Cybenko (1989); Hornik et al. (1989); Hornik (1993). The input-to-output mapping of this class of networks can be expressed, for m hidden neurons, as:

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{m} w_i \phi_{\text{MLP}}(\mathbf{u}_i, \mathbf{x}) \;, \tag{5.1}$$

where $\{w_i\}_{i=1}^m$ are the output weights and $\phi(\mathbf{u}, \mathbf{x})$ is a sigmoid activation function, which in turn depends on input weights $\{\mathbf{u}_i\}_{i=1}^m$. The hyperbolic tangent is probably the most frequently used activation function in classical texts, but any other sigmoid function can be used. In our case, for analytical tractability, we will use

$$\phi_{\text{MLP}}(\mathbf{u}, \mathbf{x}) = \text{erf}(\mathbf{u}^\top \tilde{\mathbf{x}}) = \text{erf}\left(u_0 + \sum_{d=1}^{D} u_d x_d\right) \;, \tag{5.2}$$

where $\tilde{\mathbf{x}} = [1,\ \mathbf{x}^\top]^\top$ is the augmented input vector and u_0 is regarded as the "bias" term of the neuron. The error function is in turn defined as:

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t \;. \tag{5.3}$$

As we know, networks of this type are traditionally trained by selecting their weights $\{w_i, \mathbf{u}_i\}_{i=1}^m$ so as to minimize the squared error over the training set, i.e. $\sum_{j=1}^n (f(\mathbf{x_j}) - y_j)^2$. This can be done by using the widely known backpropagation algorithm from Bryson and Ho (1969) to efficiently compute the derivatives of the cost functional wrt the network weights and then applying gradient based methods (typically, stochastic gradient descent) to minimize it.

Training MLPs in this fashion has several drawbacks, as discussed at the beginning of Section 3.1 for general classical networks: Overfitting for big m, risk of convergence to poor local minima, need for cross-validation, and lack of proper probabilistic predictions. The only benefit of these models, when compared with sparse GPs, seems to be their reduced computational complexity, and this is only true when the simplest[1] types of optimization are applied (which in turn are more likely to get stuck at undesirable minima).

Some of the solutions proposed to reduce the impact of these issues address the optimization procedure rather than the cost functional being optimized. A typical example of this is "early stopping", a technique that stops the optimization process as soon as the squared error on a separate validation set stops decreasing. This idea does work to some extent to avoid overfitting, but it clearly serves two contradictory purposes: We both try to find a (local) minimum and to avoid reaching it to prevent overfitting. Having to stop the search half way to a local minimum seems to imply that we are solving the wrong optimization problem. Also, the optimization path becomes more important than the optimizer's ability to minimize the cost function, which should be the only concern.

It seems more reasonable to address this issues acting upon the MLP model itself, changing it in such a way that the minimum of the cost functional corresponds to a desirable model. One possible way to do this is to apply a Bayesian approach, as described in the next subsection.

[1] There is only a computational advantage if the optimization procedure scales sub-quadratically with the number of hidden units (this would include variants of gradient descent, but not more involved methods such as Levenberg-Marquardt).

5.1.2 MLPs in the infinite limit

A principled and elegant solution that avoids the previously mentioned drawbacks is to resort to a non-parametric, Bayesian approach: Use an infinite number of hidden neurons, place a prior on all the weights of the network, and integrate them out. Difficult as it seems, it turns out that it is possible to do this analytically.

Following Neal (1996), we place independent zero-mean Gaussian priors on all the network weights in (5.1). The prior variance is set to σ_0^2/m for the output weights, to ℓ_d^{-2} for the input weights corresponding to dimension d, and to ℓ_0^{-2} for the input biases. Then we can compute the first two moments of the output of the network taking the expectation over the network weights:

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= 0 \qquad (5.4) \\ \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] &= \frac{1}{m}\sum_{i=1}^{m}\sigma_0^2\mathbb{E}[\phi_{\text{MLP}}(\mathbf{u}_i,\mathbf{x})\phi_{\text{MLP}}(\mathbf{u}_i,\mathbf{x}')] \\ &= \sigma_0^2\mathbb{E}[\phi_{\text{MLP}}(\mathbf{u},\mathbf{x})\phi_{\text{MLP}}(\mathbf{u},\mathbf{x}')] \\ &= \frac{2\sigma_0^2}{\pi}\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^\top\tilde{\mathbf{L}}^{-2}\tilde{\mathbf{x}}'}{\sqrt{1+2\tilde{\mathbf{x}}^\top\tilde{\mathbf{L}}^{-2}\tilde{\mathbf{x}}}\sqrt{1+2\tilde{\mathbf{x}}'^\top\tilde{\mathbf{L}}^{-2}\tilde{\mathbf{x}}'}}\right), \qquad (5.5) \end{aligned}$$

where $\tilde{\mathbf{L}} = \text{diag}([\ell_0,\ \ell_1,\ldots,\ \ell_D])$ is a diagonal matrix containing the length-scales and the bias prior ℓ_0^2.

Result (5.4) follows trivially, since all output weights $\{w_i\}_{i=1}^m$ are zero mean and independent from the input weights. Similarly, the first step of the second expectation follows because all input weights $\{\mathbf{u}_i\}_{i=1}^m$ are i.i.d., drawn from $\mathcal{N}(\mathbf{u}|\mathbf{0},\tilde{\mathbf{L}}^{-2})$, but the second step is quite involved, see Williams (1997) for the complete proof.

Since the activation function is bounded, all the moments of $f(\mathbf{x})$ are bounded, and due to the Central Limit theorem, when the number of hidden units m tends to infinity, $f(\mathbf{x})$ converges to a zero-mean GP with covariance function (5.5). We will refer to it as the ARD MLP covariance function, since regarding length-scales $\{\ell_d\}_{d=1}^D$ as hyperparameters provides the same Automatic Relevance Detection (ARD) capabilities described for ARD SE in Section 1.1.2.

It is possible to use this infinite MLP to make predictions with finite computation by invoking eq. (1.15), treating it as a GP and enjoying all its advantages. It does not come as a surprise that the performance of this infinite MLP is very good, given that it is a

combination of *all possible networks* (with some non-zero a priori probability) having an *infinite number of hidden neurons*. However, since this is a full GP, computational complexity has now been raised to $\mathcal{O}(n^3)$, making its use impractical for big data sets.

To workaround this, it is possible to use the SPGP approximation with covariance function (5.5). This yields a sparse GP that approximates an MLP, though the original MLP structure disappears: Predictions are now a combination of arcsine functions instead of sigmoids.

5.1.3 Marginalized MLPs (MMLPs)

Instead of integrating out all the parameters of an MLP and then sparsifying the resulting model, we can follow the MN approach and integrate out *only the output* weights. Predictive and NLML equations are then given by (3.4) and (3.5), with $\phi_{\text{MLP}}(\mathbf{u}, \mathbf{x})$ being the basis function family.

Constant σ_{p}^2 needed for both equations can be derived from the family of basis functions

$$\sigma_{\text{p}}^2 = \lim_{L\to\infty} \frac{1}{L^D} \int_{C_L} \text{erf}^2(\mathbf{u}_i^\top \tilde{\mathbf{x}}) \text{d}\mathbf{x} = 1, \tag{5.6}$$

where region C_L is a cube of edge L, centered at the coordinate origin.

The computational and storage requirements of MMLPs are the same as for any other MN, as described towards the end of Section 3.1.1.

As usual, we reparametrize each input weight $\mathbf{u}_i$ as $\mathbf{u}_i = \tilde{\mathbf{L}}^{-1}\boldsymbol{\omega}_i$, so that the components of any given dimension can be scaled at once for all input weights, using the corresponding length-scale. With this definition, to completely define an MMLP we must specify σ_0^2, σ^2, $\{\ell\}_{d=0}^D$ and $\{\boldsymbol{\omega}_i\}_{i=1}^m$.

We know from Chapter 3 that MNs are zero-mean sparse GPs with covariance function (3.3). Therefore, the covariance function of MMLPs can be expressed as

$$k_{\text{MMLP}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_0^2}{m} \boldsymbol{\phi}_{\text{MLP}}(\mathbf{x})^\top \boldsymbol{\phi}_{\text{MLP}}(\mathbf{x}') = \sigma_0^2 \frac{1}{m} \sum_{i=1}^m \phi_{\text{MLP}}(\mathbf{u}_i, \mathbf{x}) \phi_{\text{MLP}}(\mathbf{u}_i, \mathbf{x}') \ ,$$

which, apart from the scaling with σ_0^2, is the average of $\phi_{\text{MLP}}(\mathbf{u}, \mathbf{x})\phi_{\text{MLP}}(\mathbf{u}, \mathbf{x}')$ over the values of input weights $\mathbf{u} = \{\mathbf{u}_i\}_{i=1}^m$. If the input weights are distributed according to $\mathcal{N}(\mathbf{u}|\mathbf{0}, \tilde{\mathbf{L}}^{-2})$, this average is a Monte Carlo approximation of integral

$$k_{\text{MMLP}}(\mathbf{x}, \mathbf{x}') \approx \sigma_0^2 \int_{\mathbb{R}^D} \phi_{\text{MLP}}(\mathbf{u}, \mathbf{x}) \phi_{\text{MLP}}(\mathbf{u}, \mathbf{x}') \mathcal{N}(\mathbf{u}|\mathbf{0}, \tilde{\mathbf{L}}^{-2}) \text{d}\mathbf{u} \ , \tag{5.7}$$

which in turn is exactly ARD MLP covariance function (5.5) (the integral can be re-expressed as the expectation in that equation). The Monte Carlo approximation becomes exact in the limit when the number of samples m tends to infinity.

Thus, we have proved that MMLPs approximate full GPs with ARD MLP covariance function if the weights are distributed according to $\mathcal{N}(\mathbf{u}|\mathbf{0}, \tilde{\mathbf{L}}^{-2})$, or equivalently, if $\{\boldsymbol{\omega}_i\}_{i=1}^m$ are distributed according to $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_{D+1})$, with an exact correspondence with the full GP when the number of basis functions is infinite.

The SPGP with ARD MLP covariance function, though different in nature, also converges to the same full GP if the set of pseudo-inputs is a superset of the set of actual input values in the data set.

Following Chapter 3, we can define MMLP-fixed as the algorithm that lets auxiliary vectors $\{\boldsymbol{\omega}_i\}_{i=1}^m$ fixed to values drawn from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_{D+1})$ and learns the remaining $D + 3$ hyperparameters. Such a method would present no overfitting, but would also have poor performance for small m (especially on high-dimensional problems, since the volume of the space sampled by the Monte Carlo approximation grows exponentially with the dimension of input data). As we also know from Chapter 3, directly learning the input weights of MNs (such as MMLPs) may distort their distribution and produce overfitting. Therefore, in the following, we will consider the improved MN versions proposed in Sections 3.2 (noise bounding trick) and 3.3 (network mixing).

5.1.4 Bounded-Noise Marginalized MLPs (BN-MMLPs)

Applying the noise bounding trick described in Section 3.2 to MMLPs yields the same algorithm used for BN-MCN, but with (5.2) as basis function family and $\sigma_{\text{p}}^2 = 1$ as per (5.6). Also recall that $\mathbf{u}_i = \tilde{\mathbf{L}}^{-1}\boldsymbol{\omega}_i$. In detail, the procedure is:

1. Run MMLP-fixed:

   - Initialize $\{\ell_d\}_{d=1}^D$, ℓ_0, σ_0^2, and σ^2 to some sensible values. (We will use: One half of the ranges of the input dimensions, $\sqrt{\sum_{d=1}^D \ell_D^2}$, the variance of the outputs $\{y_j\}_{j=1}^n$ and $\sigma_0^2/4$, respectively).
   - Fix $\{\boldsymbol{\omega}_i\}_{i=1}^m$ to random values drawn from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \mathbf{I}_{D+1})$ (to approximate the ARD MLP covariance function).

- Minimize (3.5), the NLML of the model, wrt to σ_0^2, σ^2, and $\{\ell\}_{d=1}^D$ (keeping $\{\boldsymbol{\omega}\}_{i=1}^m$ fixed).

2. Run BN-MMLP:

   - Initialize $\{\ell\}_{d=0}^D$, σ_0^2, σ^2 and $\{\boldsymbol{\omega}_i\}_{i=1}^m$ to the values they had after MMLP-fixed converged.
   - Set σ_{min}^2 to the value found for σ^2 after convergence of MMLP-fixed. Initialize σ^2 slightly above σ_{min}^2. (We will use $1.5\sigma_{\text{min}}^2$).
   - Minimize (3.5), the NLML of the model, wrt to $\{\ell_d\}_{d=0}^D$, σ_0^2, σ^2 and $\{\boldsymbol{\omega}_i\}_{i=1}^m$ with the constraint $\sigma^2 > \sigma_{\text{min}}^2$.

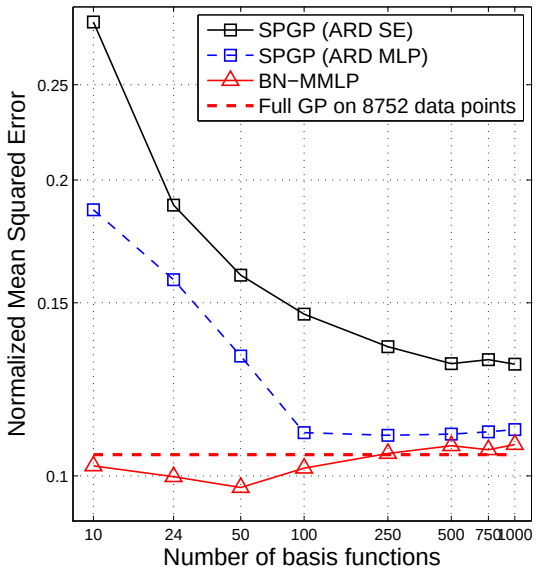
As usual, the required minimizations will be performed by conjugate gradient descent, using the analytic derivatives of the NLML. The expressions of these derivatives are provided in Section E.2 of Appendix E.

5.1.4.1 Experiments

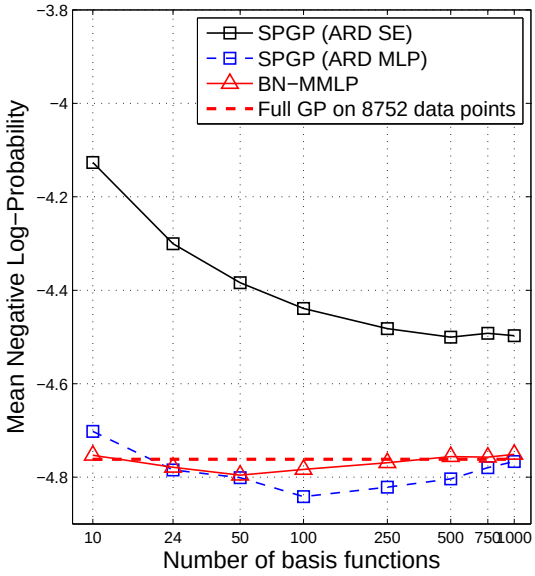
Here we reproduce the experiments of Section 3.2.4, using exactly the same settings, but using BN-MMLP instead of BN-MCN. Also, in addition to our traditional benchmark (SPGP with ARD SE covariance function), we provide results for SPGP with ARD MLP covariance function (5.5). Since this latter covariance function is not translation invariant, we additionally center all input data.

Results for *Elevators* and *Pole Telecomm* are shown in Fig. 5.1 and 5.2, respectively. If we consider the NMSE error measure, BN-MMLP performs better than SPGP with the ARD SE covariance function in both problems. However, when the ARD MLP version is used, SPGP manages to beat BN-MMLP in the *Pole Telecomm* problem. As expected, BN-MMLP does not perform specially well in the MNLP sense, being beaten by both versions of SPGP in the *Pole Telecomm* problem.

Figs. 5.3 and 5.4 show results for data sets *Kin-40k* and *Pumady-32nm*. In both problems, BN-MLP produces the best results in the NMSE sense (even outperforming the full GP for a modest number of basis functions). For *Kin-40k*, the ARD MLP version of SPGP hardly provides any advantage over the more typical ARD SE covariance function. On the other hand, for *Pumady-32nm*, the ARD MLP roughly halves

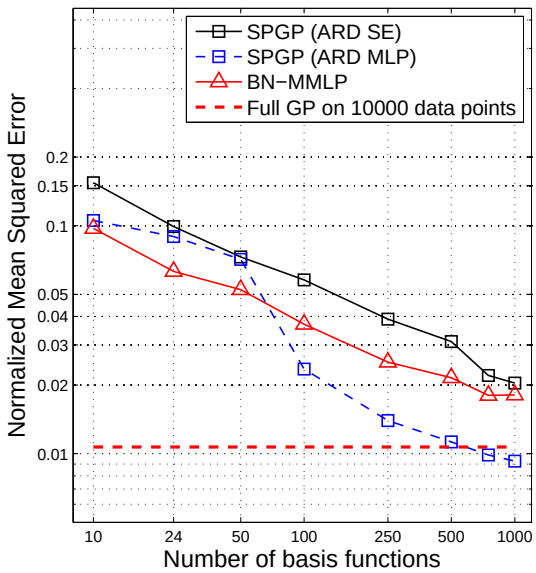


(a) NMSE (log-log plot)

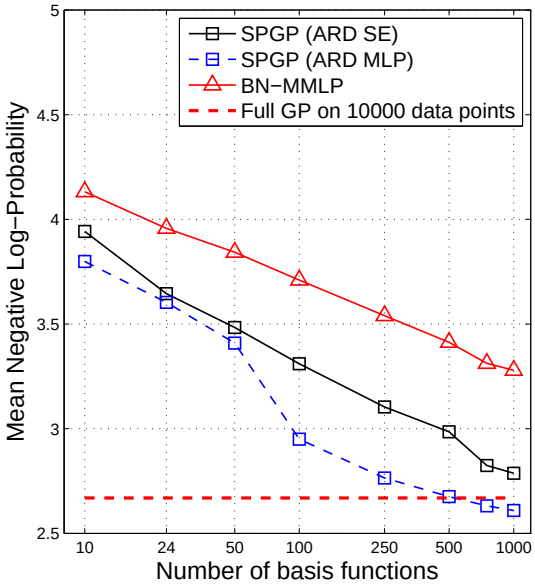


(b) MNLP (semilog plot)

Figure 5.1: NMSE and MNLP for SPGP (with both ARD SE and ARD MLP cov. functions), BN-MMLP and full GP, for the *Elevators* problem.

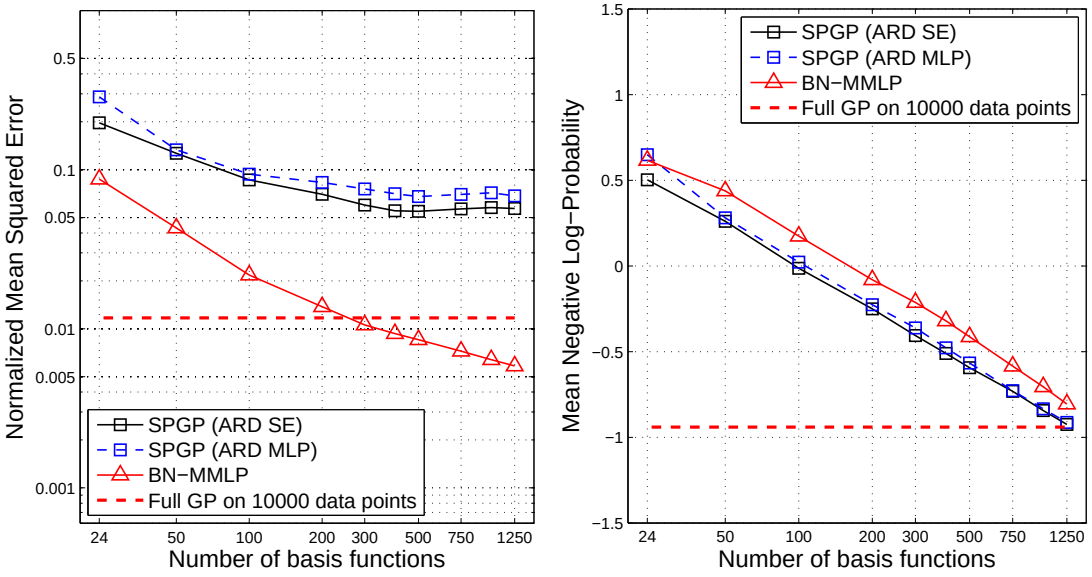


(a) NMSE (log-log plot)



(b) MNLP (semilog plot)

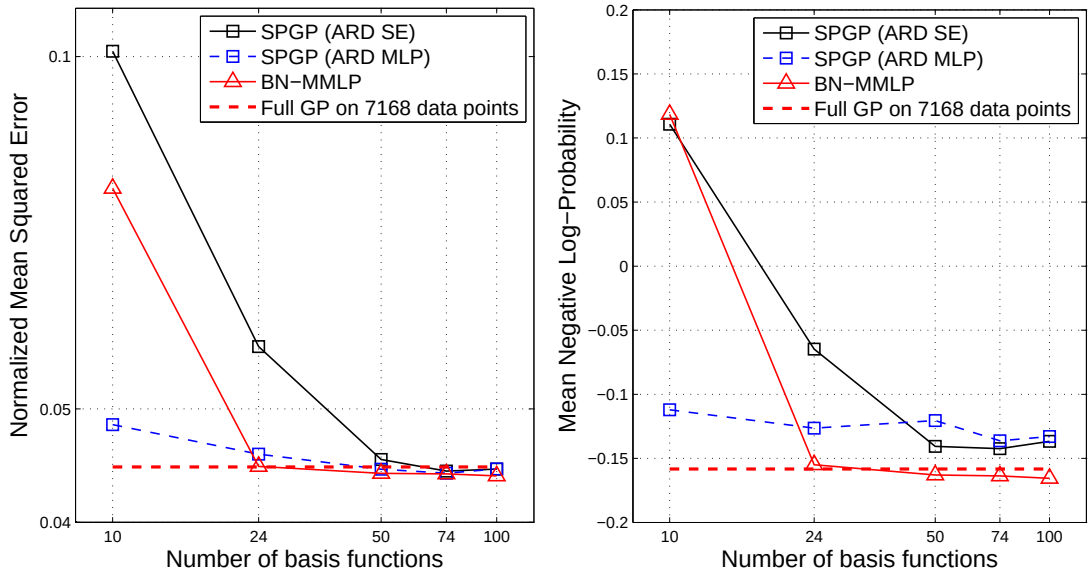
Figure 5.2: NMSE and MNLP for SPGP (with both ARD SE and ARD MLP cov. functions), BN-MMLP and full GP, for the *Pole Telecomm* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 5.3: NMSE and MNLP for SPGP (with both ARD SE and ARD MLP cov. functions), BN-MMLP and full GP, for the *Kin-40k* problem.



(a) NMSE (log-log plot)

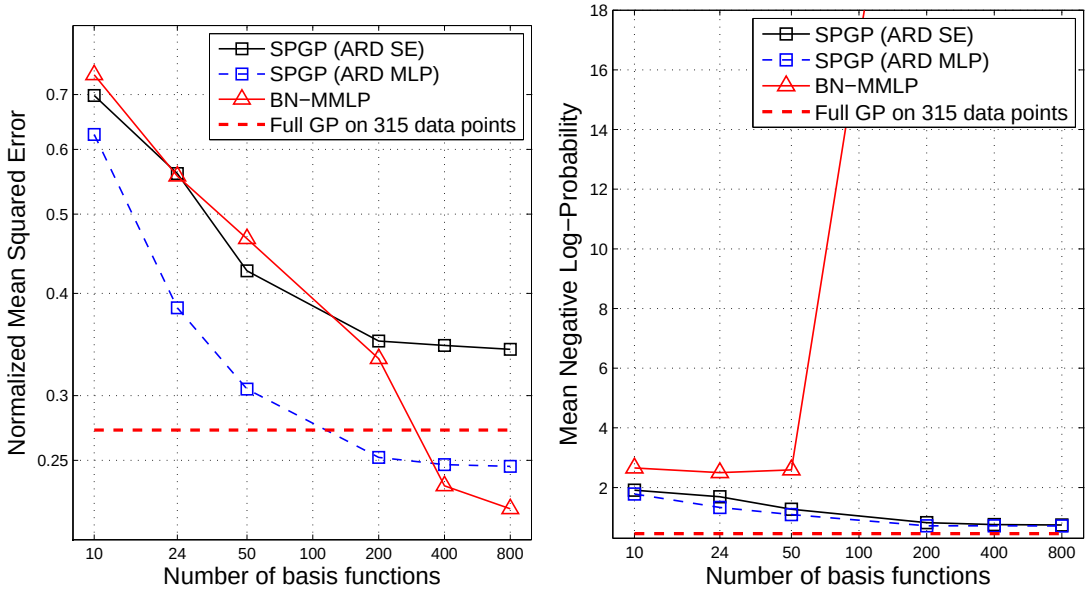
(b) MNLP (semilog plot)

Figure 5.4: NMSE and MNLP for SPGP (with both ARD SE and ARD MLP cov. functions), BN-MMLP and full GP, for the *Pumadyn-32nm* problem.

the NMSE when m is small. As usual, all methods correctly determine the relevant dimensions in the *Pumady-32nm* problem to be $[4,\ 5,\ 15,\ 16]$.

Results for our last data set, *Pendulum*, are shown in Fig. 5.5. We can confirm the expected behavior: Though predictive means are accurate, even outperforming the full GP when the number of basis functions is very high[2], predictive variances are exceedingly small, resulting in very poor values for the MNLP measure.

The performance of SPGP in this data set is again boosted when the ARD MLP covariance function is used, with this sparse method outperforming a full GP with ARD SE covariance function for $m \geq 200$.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 5.5: NMSE and MNLP for SPGP (with both ARD SE and ARD MLP cov. functions), BN-MMLP and full GP, for the *Pendulum* problem.

5.1.5 Marginalized MLP Mixture (MMLPmix)

Combining several MMLPs as described in Section 3.3 improves their robustness. The resulting algorithm, MMLPmix, is identical to MCNmix except for the basis function

[2] As we know, sparsity and computational advantages are lost when $m > n$, but the approach may still be valid to obtain highly accurate predictions on some small data sets.

family and normalizing constant, which are now (5.2) and $\sigma_{\mathrm{p}}^2 = 1$, respectively. Recall that the input weights are defined as $\mathbf{u}_i = \tilde{\mathbf{L}}^{-1}\boldsymbol{\omega}_i$. The procedure is:

1. Train K different MMLPs (using random initialization, as described in Section 5.1.4 for MMLP-fixed), minimizing (3.5) wrt to $\{\ell_d\}_{d=0}^D$, σ_0^2, σ^2 and $\{\boldsymbol{\omega}_i\}_{i=1}^m$.
2. Use the K MMLPs to obtain the K predictive means and variances at new test points.
3. Combine them using equation 3.8.

Analytic derivatives wrt all hyperparameters are provided in Section E.2 of Appendix E, so that conjugate gradient descent can be used. Time and storage space needs are identical to those of any other MN mixture, as described in Section 3.3.2.

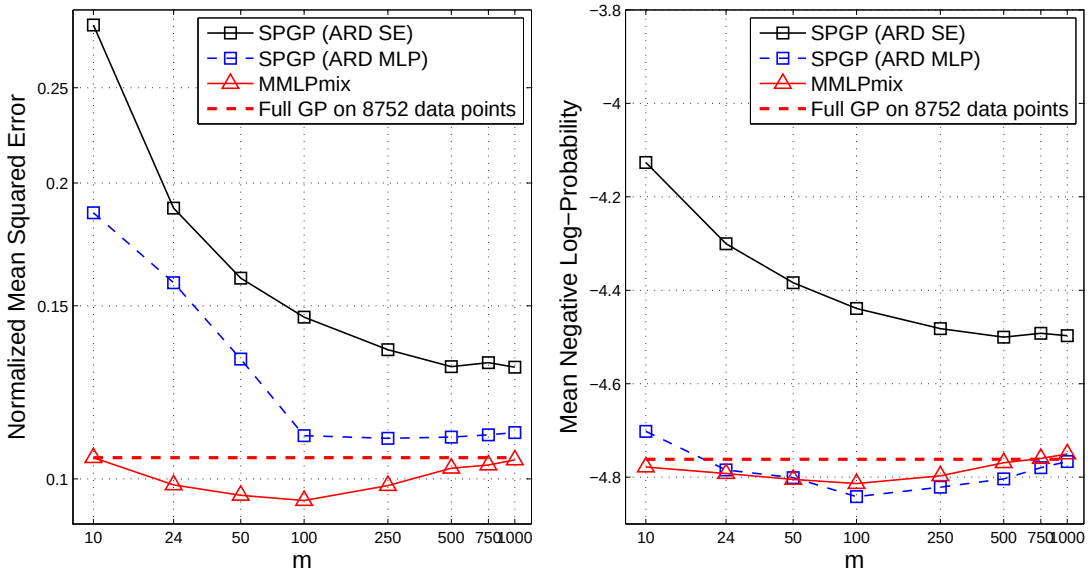
5.1.5.1 Experiments

Now we will reproduce the experiments of Section 3.3.3, with identical setup, but using MMLPmix instead of MCNmix. As before, we also include results for SPGP with ARD MLP covariance function and center data before running the experiments (SPGP with the ARD MLP covariance function is the only method affected by a translation of the inputs).

We display results for *Elevators* and *Pole Telecomm* in Figs. 5.6 and 5.7 respectively. MMLPmix gets surpassed by SPGP (using the ARD MLP covariance function) in *Pole Telecomm*, but only for big m, and only in the MNLP measure. Overall, MMLPmix produces the best results.

If we turn to *Kin-40k* and *Pumadyn-32nm*, displayed in Figs. 5.8 and 5.9, we see MMLPmix is the clear winner. For *Kin-40k*, it does not only beat the full GP in NMSE (which also was achieved by BN-MMLP), but also in MNLP. For *Pumadyn-32nm*, it quickly reaches full GP performance, closely followed by SPGP. All methods select the correct set of relevant inputs, $[4,\ 5,\ 15,\ 16]$.

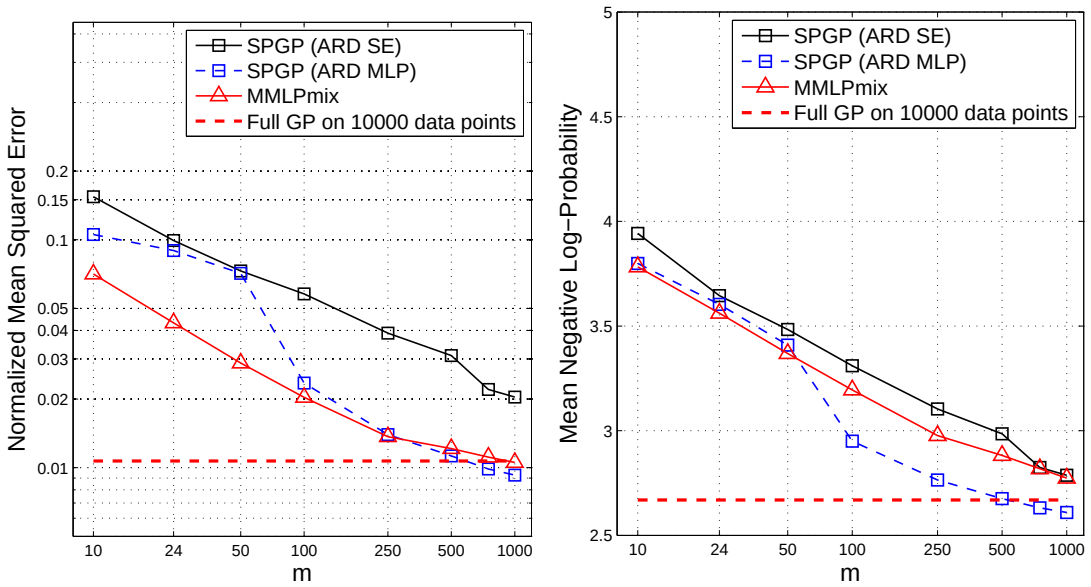
Finally, we can see how mixing as little as four networks considerably improves predictive variances. Fig. 5.10 shows results for the *Pendulum* data set, where a significant improvement in MNLP can be appreciated with respect to Fig. 5.5, where noise bounding was applied.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

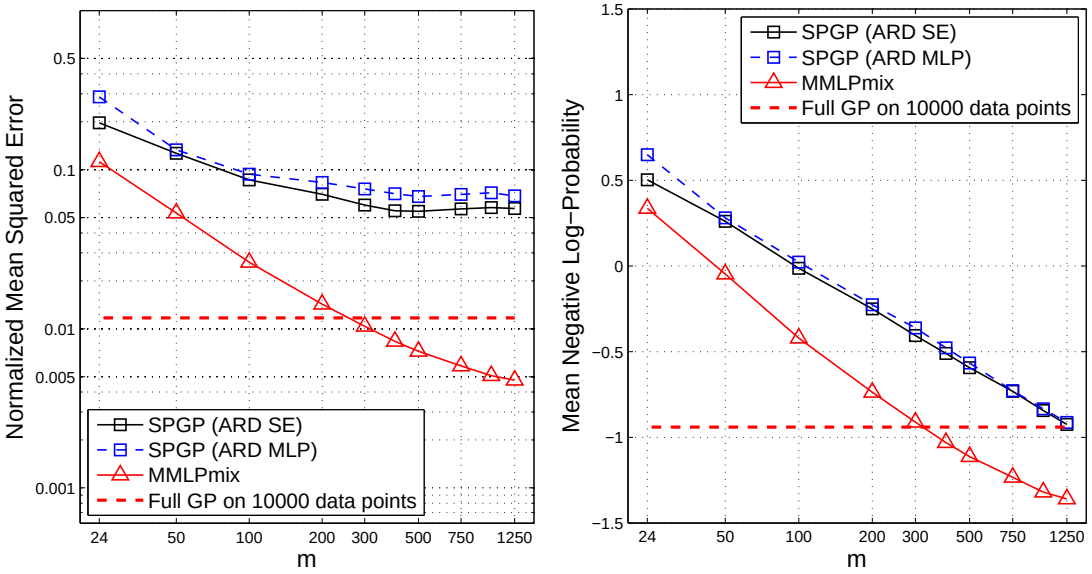
Figure 5.6: NMSE and MNLP for SPGP (ARD SE/ARD MLP, m basis functions), MMLPmix (4 networks, $m/2$ bases each) and full GP, for the *Elevators* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

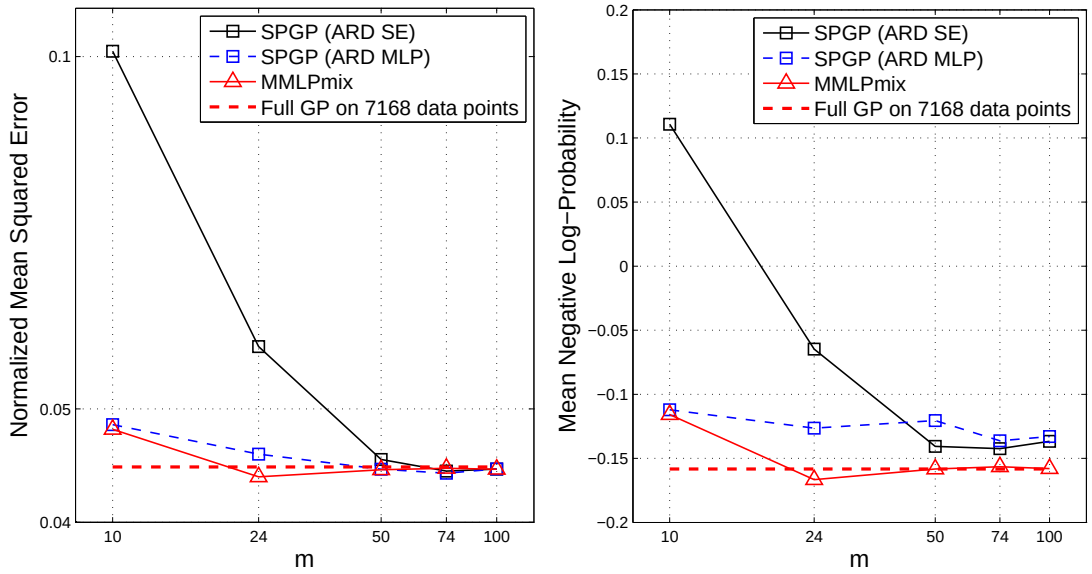
Figure 5.7: NMSE and MNLP for SPGP (ARD SE/ARD MLP, m basis functions), MMLPmix (4 networks, $m/2$ bases each) and full GP, for the *Pole Telecomm* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

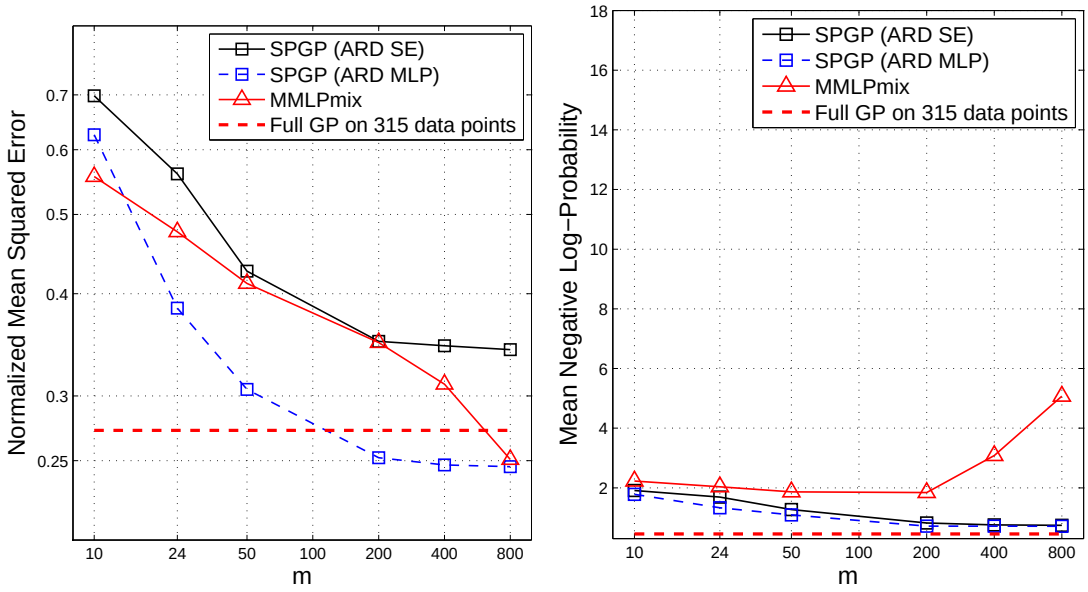
Figure 5.8: NMSE and MNLP for SPGP (ARD SE/ARD MLP, m basis functions), MMLPmix (4 networks, $m/2$ bases each) and full GP, for the *Kin-40k* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 5.9: NMSE and MNLP for SPGP (ARD SE/ARD MLP, m basis functions), MMLPmix (4 networks, $m/2$ bases each) and full GP, for the *Pumadyn-32nm* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 5.10: NMSE and MNLP for SPGP (ARD SE/ARD MLP, m basis functions), MMLPmix (4 networks, $m/2$ bases each) and full GP, for the *Pendulum* problem.

5.1.6 Discussion

In the preceding sections, we have presented results of regression experiments with different types of sparse MLPs (BN-MMLP, MMLPmix and the SPGP with ARD MLP covariance function). After careful inspection, and comparing with the results of Sections 3.2.4 and 3.3.3, where BN-MCN and MCNmix were put to test, we can draw the following conclusions:

(a) Despite almost every mention of SPGP in the literature is tied to the ARD SE covariance function, the ARD MLP covariance function typically yields better performance in the sparser regime.

(b) The predictive means obtained by BN-MMLP and MMLPmix are generally better than those obtained by SPGP.

(c) BN-MMLP's predictive variances are reasonable for most problems, but occasionally they may be exceedingly small. Thus, one should watch out for overconfident

predictions when using BN-MMLP. In MMLPmix, this problem is greatly reduced. SPGP does not seem to ever present this problem and is therefore a safer method.

(d) The functional form of the predictive mean for BN-MMLP and MMLPmix corresponds exactly to the structure of an MLP, so both methods can be used to train traditional MLPs. This is not possible with SPGP.

(e) The choice of the activation function (i.e., the definition of $\phi(\mathbf{u}, \mathbf{x})$) matters, but not critically. Comparing these results with those obtained with BN-MCN and MCNmix, we see that both cosine and sigmoid bases (which are functionally very different) display similar behavior and performance.

5.2 Non-Gaussian likelihoods

We have so far considered inference and model selection in sparse models with Gaussian likelihood, i.e. $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I}_n)$. In this section we will detail how all these models can be extended to handle non-Gaussian likelihoods by resorting to approximate inference. As we will see in the next two sections, being able to use other types of likelihoods enables us to use these models for robust regression and classification.

Two different schemes for approximate inference in GP models with non-Gaussian likelihoods are typically used: The simple Laplace approximation of Williams and Barber (1998) and the more sophisticated Expectation Propagation (EP) of Minka (2001). According to the work by Kuss (2006), the Laplace approximation is not well suited to non-derivable likelihoods (such as the one we will be using for robust regression) and produces less accurate posterior distributions than EP. The superiority of EP over the Laplace approximation is extensively discussed in Kuss and Rasmussen (2005) and Kuss and Rasmussen (2006), so it will be our tool of choice.

5.2.1 Expectation Propagation (EP)

Expectation Propagation (EP) is a technique for approximate inference in Bayesian models. It was introduced by Minka (2001), and it can be applied to a variety of settings. Here we will limit ourselves to describe it for the case of Bayesian models

with GP prior and factorizing likelihood. We follow Kuss (2006) and Rasmussen and Williams (2006), where further information about applying EP to GPs can be found.

First, let us introduce the so-called "natural parametrization" $\mathcal{N}_{\text{nat}}(\mathbf{f}|\boldsymbol{\nu}, \mathbf{T})$ for the Gaussian distribution, since some of the expressions below are simpler or more easy to compute when considering this form:

$$\mathcal{N}_{\text{nat}}(\mathbf{f}|\boldsymbol{\nu}, \mathbf{T}) = \left|\frac{\mathbf{T}}{2\pi}\right|^{\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{\nu}^\top \mathbf{T}^{-1}\boldsymbol{\nu}\right) \exp\left(-\frac{1}{2}\mathbf{f}^\top \mathbf{T}\mathbf{f} + \boldsymbol{\nu}^\top \mathbf{f}\right) ,$$

where $\mathbf{T}$ is usually referred to as the "precision matrix". We will combine this notation with the usual one in terms of the mean and the covariance. Both are simply related:

$$\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}_{\text{nat}}(\mathbf{f}|\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}) , \quad \mathcal{N}_{\text{nat}}(\mathbf{f}|\boldsymbol{\nu}, \mathbf{T}) = \mathcal{N}(\mathbf{f}|\mathbf{T}^{-1}\boldsymbol{\nu}, \mathbf{T}^{-1}) . \tag{5.8}$$

Using this notation, we can express a general GP model as

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}_{\text{nat}}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{\mathbf{ff}}^{-1}) , \qquad p(\mathbf{y}|\mathbf{f}) = \prod_{j=1}^{n} p(y_j|f_j)$$

where the functional form of $p(y_j|f_j)$ will determine whether it is a Gaussian-noise regression model, a robust regression model, a classification model, etc. The likelihood factorizes because each observation is assumed to depend only on its corresponding latent value.

For any given likelihood, it is possible to express the posterior over latent values $\mathbf{f}$ using Bayes formula (1.9):

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{1}{Z} p(\mathbf{f}|\mathbf{X}) \prod_{j=1}^{n} p(y_j|f_j) , \tag{5.9}$$

where Z is the evidence of the model:

$$Z = p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{f}|\mathbf{X}) \prod_{j=1}^{n} p(y_j|f_j) \mathrm{d}\mathbf{f} . \tag{5.10}$$

However, if likelihood $p(y_j|f_j)$ is not Gaussian, the key equations for model selection and inference become intractable: Evidence (5.10) cannot be computed analytically (which in turn means that it is not possible to use ML-II for model selection) and $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ becomes non-Gaussian, so that it is no longer possible to obtain analytical expressions for the marginals or the predictive distributions.

5.2.1.1 Approximate marginal posterior

To overcome these problems, EP seeks to replace the true posterior $p(\mathbf{f}|\mathbf{X},\mathbf{y})$ by an approximate posterior $q(\mathbf{f}|\mathbf{X},\mathbf{y})$ which is Gaussian. In order to do this, the following approximation is introduced:

$$p(y_j|f_j) \approx t(f_j,\tilde{\nu}_j,\tilde{\tau}_j,\tilde{Z}_j) \equiv \tilde{Z}_j \mathcal{N}_{\text{nat}}(f_j|\tilde{\nu}_j,\tilde{\tau}_j) \;, \tag{5.11}$$

where $t_j(f_j) = t(f_j,\tilde{\nu}_j,\tilde{\tau}_j,\tilde{Z}_j)$ are called the *site functions* and $\{\tilde{\nu}_j,\tilde{\tau}_j,\tilde{Z}_j\}_{j=1}^n$ are the site parameters. The site functions are unnormalized, unidimensional Gaussians in f_j that locally approximate the likelihood. For notational convenience, we will omit the dependence of the site functions on the site parameters and use simply $t_j(f_j)$ to denote them. Using (5.11), the joint likelihood can be expressed as

$$\prod_{j=1}^n p(y_j|f_j) \approx \prod_{j=1}^n t_j(f_j) = \mathcal{N}_{\text{nat}}(\mathbf{f}|\tilde{\boldsymbol{\nu}},\tilde{\mathbf{T}}) \prod_{j=1}^n \tilde{Z}_j \tag{5.12}$$

where $\tilde{\boldsymbol{\nu}} = [\tilde{\nu}_1,\ldots,\tilde{\nu}_n]^\top$ and $\tilde{\mathbf{T}} = \text{diag}(\tilde{\tau}_1,\ldots,\tilde{\tau}_n)$ collect the natural parameters of all sites.

If we assume the site parameters as given, the approximate posterior over the latent values can be computed by inserting (5.11) into (5.9) and (5.10):

$$p(\mathbf{f}|\mathbf{X},\mathbf{y}) \approx q(\mathbf{f}|\mathbf{X},\mathbf{y}) = \frac{1}{Z_{\text{EP}}} p(\mathbf{f}|\mathbf{X}) \prod_{j=1}^n t_j(f_j) = \mathcal{N}_{\text{nat}}(\mathbf{f}|\boldsymbol{\nu},\mathbf{T}) \tag{5.13}$$

$$\text{with} \quad \boldsymbol{\nu} = \mathbf{0} + \tilde{\boldsymbol{\nu}} \;, \quad \mathbf{T} = \mathbf{K}_{\mathbf{ff}}^{-1} + \tilde{\mathbf{T}} \;, \quad \text{and}$$

$$Z_{\text{EP}} = q(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{f}|\mathbf{X}) \prod_{j=1}^n t_j(f_j) \text{d}\mathbf{f} \;. \tag{5.14}$$

Since (5.13) is the product of two Gaussians, it is also a Gaussian (correct normalization is ensured by the constant Z_{EP}, which corresponds to the approximate evidence). The natural parameters of the posterior are straightforward to compute, since they are the sum of those of the multiplying Gaussians. We emphasize this by explicitly including natural parameter $\mathbf{0}$ of the prior. Note that we use tilde variables to refer to the site parameters and non-tilde variables to refer to the posterior parameters.

To obtain the posterior marginal distributions of each latent variable, it is more convenient to rewrite (5.13) in the mean-covariance form $q(\mathbf{f}|\mathbf{X},\mathbf{y}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu},\boldsymbol{\Sigma})$.

Using the equivalence provided by (5.8) and the marginalization property (B.2), the posterior mean and covariance matrix, as well as the marginals, are:

$$\boldsymbol{\Sigma} = (\mathbf{K}_{\mathbf{ff}}^{-1} + \tilde{\mathbf{T}})^{-1} \text{ with marginals } \sigma_j^2 = [\boldsymbol{\Sigma}]_{jj} \tag{5.15a}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\tilde{\boldsymbol{\nu}} \text{ with marginals } \mu_j = [\boldsymbol{\mu}]_j \, . \tag{5.15b}$$

5.2.1.2 The cavity distribution

Another way to compute the approximate marginal posterior distribution is to start from the approximate joint posterior and integrate out every other variable

$$q(f_j|\mathbf{X},\mathbf{y}) = \int q(\mathbf{f}|\mathbf{X},\mathbf{y})\mathrm{d}\mathbf{f}^{\backslash j} = \frac{1}{Z_{\text{EP}}} \int p(\mathbf{f}|\mathbf{X}) \prod_{q=1}^{n} t_q(f_q)\mathrm{d}\mathbf{f}^{\backslash j} \tag{5.16}$$

where we use $\mathbf{f}^{\backslash j}$ to refer to all latent values except f_j. Since the j-th term of the product in (5.16), $t_j(f_j)$, does not depend on $\mathbf{f}^{\backslash j}$, it can be taken out of the integral, and we can express the marginal posteriors as

$$q(f_j|\mathbf{X},\mathbf{y}) = \frac{1}{Z_{\text{EP}}} t_j(f_j) q_{\backslash j}(f_j) \quad \text{with} \quad q_{\backslash j}(f_j) = \int p(\mathbf{f}|\mathbf{X}) \prod_{q\neq j}^{n} t_q(f_q)\mathrm{d}\mathbf{f}^{\backslash j} \tag{5.17}$$

where $q_{\backslash j}(f_j)$ is known as the *cavity distribution*, and it has the form of an unnormalized Gaussian, $q_{\backslash j}(f_j) \propto \mathcal{N}_{\text{nat}}(f_j|\nu_{\backslash j}, \tau_{\backslash j})$. The cavity distribution corresponds (up to a scaling) to removing the approximate likelihood j from the joint posterior. Since marginal posteriors are known from (5.15), we have that $q(f_j|\mathbf{X},\mathbf{y}) = \mathcal{N}(f_j|\mu_j, \sigma_j^2)$, so that

$$t_j(f_j) q_{\backslash j}(f_j) \propto \mathcal{N}(f_j|\mu_j, \sigma_j^2) \tag{5.18}$$

Again, we have the product of two Gaussians, which is proportional to a Gaussian. Since site parameters and posterior marginals are known, we can compute the cavity parameters:

$$\nu_{\backslash j} = \sigma_j^{-2}\mu_j - \tilde{\nu}_j \tag{5.19a}$$

$$\tau_{\backslash j} = \sigma_j^{-2} - \tilde{\tau}_j \tag{5.19b}$$

Despite the form of (5.19), cavity parameters $\nu_{\backslash j}$ and $\tau_{\backslash j}$ are independent of site parameters $\tilde{\nu}_j$ and $\tilde{\tau}_j$, as the very definition of $q_{\backslash j}(f_j)$, (5.17), states.

So, given the prior and the site parameters, it is possible to compute the approximate marginals, the approximate evidence, and the cavity parameters. But how do we determine the site parameters?

5.2.1.3 Obtaining the site parameters

The key idea of EP is to select the site parameters so that the product of the site function (the approximate likelihood) and the cavity distribution is as close as possible to the product of the *true likelihood* $p(y_j|f_j)$ and the cavity distribution:

$$t(f_j, \tilde{\nu}_j, \tilde{\tau}_j, \tilde{Z}_j)\mathcal{N}_{\text{nat}}(f_j|\nu_{\setminus j}, \tau_{\setminus j}) \approx p(y_j|f_j)\mathcal{N}_{\text{nat}}(f_j|\nu_{\setminus j}, \tau_{\setminus j}) \tag{5.20}$$

The left hand side of (5.20) is, up to a constant, the approximate marginal posterior $q(f_j|\mathbf{X}, \mathbf{y})$, which is Gaussian. The right hand side is known as the *tilted distribution* and is not Gaussian in general. For both distributions to be as close as possible (in the sense of minimizing the Kullback-Leibler divergence), the j-th site parameters must be chosen so that the first and second moments of both distributions are matched. To ensure proper normalization, also the zeroth moment must be matched.

In order to do this, the moments of the tilted distribution must be computed. The concrete expressions will depend on the functional form of the likelihood. A general technique to compute them is to use the moment generating function

$$M_j(\lambda) = \int \exp(\lambda f_j) p(y_j|f_j)\mathcal{N}_{\text{nat}}(f_j|\nu_{\setminus j}, \tau_{\setminus j})\mathrm{d}f_j \tag{5.21}$$

and then evaluate the derivatives wrt λ at zero to obtain the non-central moments

$$m_{0j} = M_j(0)\ , \quad m_{kj} = \frac{1}{m_{0j}} \frac{\partial M_j(\lambda)}{\partial \lambda}\bigg|_{\lambda=0} \quad \forall\, k \in \mathbb{N} : k > 0\ , \tag{5.22}$$

as described, for instance, in DeGroot and Schervish (2002). Matching these moments with those of the left hand side of (5.20) and solving for the site parameters, yields:

$$\tilde{\tau}_j = (m_{2j} - m_{1j}^2)^{-1} - \tau_{\setminus j} \tag{5.23a}$$

$$\tilde{\nu}_j = m_{1j}(\tau_{\setminus j} + \tilde{\tau}_j) - \nu_{\setminus j} \tag{5.23b}$$

$$\tilde{Z}_j = m_{0j}\sqrt{2\pi(\tau_{\setminus j}^{-1} + \tilde{\tau}_j^{-1})} \exp\left(\frac{(\tau_{\setminus j}^{-1}\nu_{\setminus j} - \tilde{\tau}_j^{-1}\tilde{\nu}_j)^2}{2(\tau_{\setminus j}^{-1} + \tilde{\tau}_j^{-1})}\right) \tag{5.23c}$$

(recall that the j-th cavity parameters were already known and independent of the j-th site parameters). Of course, the expressions describing m_{kj} will be different for each possible likelihood function and depend on parameters y_j, $\nu_{\setminus j}$, $\tau_{\setminus j}$.

After updating the site parameters, the posterior distribution must be updated accordingly using (5.15).

5.2.1.4 Model selection and inference

The evidence of the model is approximated by Z_{EP}, (5.14). Expanding in terms of the covariance matrix and site parameters (after EP reaches a fixed point), we have

$$-\log q(\mathbf{y}|\mathbf{X}) = -\log \int p(\mathbf{f}|\mathbf{X}) \prod_{j=1}^{n} t_j(f_j)\mathrm{d}\mathbf{f} = -\frac{1}{2}\sum_{j=1}^{n} \log \tilde{Z}_i^2 \tilde{\tau}_j + \frac{1}{2}\log\left|\tilde{\mathbf{T}}\mathbf{K}_{\mathbf{ff}} + \mathbf{I}_n\right| + \frac{1}{2}\tilde{\boldsymbol{\nu}}^\top \left(\tilde{\mathbf{T}}\mathbf{K}_{\mathbf{ff}}\tilde{\mathbf{T}} + \tilde{\mathbf{T}}\right)^{-1}\tilde{\boldsymbol{\nu}} + \frac{n}{2}\log(2\pi) \quad (5.24)$$

so that ML-II model selection can be performed by minimizing (5.24) wrt the hyper-parameters of the model.

The posterior mean of $f_* = f(\mathbf{x}_*)$, the latent function at test point $\mathbf{x}_*$, under the Gaussian approximation is

$$\begin{aligned}\mathbb{E}_q[f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*] &= \int \mathbb{E}[f_*|\mathbf{X},\mathbf{x}_*,\mathbf{f}]q(\mathbf{f}|\mathbf{X},\mathbf{y})\mathrm{d}\mathbf{f} \\ &= \int \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}\mathbf{f}\, q(\mathbf{f}|\mathbf{X},\mathbf{y})\mathrm{d}\mathbf{f} = \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}^{-1}\mathbb{E}_q[\mathbf{f}|\mathbf{X},\mathbf{y}] = \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}^{-1}\boldsymbol{\mu} \quad (5.25)\end{aligned}$$

and the variance can be expressed following Rasmussen and Williams (2006) as

$$\begin{aligned}\mathbb{V}_q[f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*] &= \mathbb{E}_{p(f_*|\mathbf{X},\mathbf{x}_*,\mathbf{f})}[(f_* - \mathbb{E}[f_*|\mathbf{X},\mathbf{x}_*,\mathbf{f}])^2] \\ &\quad + \mathbb{E}_{q(\mathbf{f}|\mathbf{X},\mathbf{y})}[(\mathbb{E}[f_*|\mathbf{X},\mathbf{x}_*,\mathbf{f}] - \mathbb{E}[f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*])^2] \\ &= k_{**} - \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}^{-1}\mathbf{k}_{\mathbf{f}*} + \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}^{-1}(\mathbf{K}_{\mathbf{ff}}^{-1} + \tilde{\mathbf{T}})^{-1}\mathbf{K}_{\mathbf{ff}}^{-1}\mathbf{k}_{\mathbf{f}*} \\ &= k_{**} - \mathbf{k}_{\mathbf{f}*}^\top (\mathbf{K}_{\mathbf{ff}} + \tilde{\mathbf{T}}^{-1})^{-1}\mathbf{k}_{\mathbf{f}*} \quad (5.26)\end{aligned}$$

so that the approximate posterior distribution of the latent variable at the new test point is

$$\begin{aligned}q(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*) &= \mathcal{N}(f_*|\, \mathbf{k}_{\mathbf{f}*}^\top \mathbf{K}_{\mathbf{ff}}^{-1}\boldsymbol{\mu},\ \ k_{**} - \mathbf{k}_{\mathbf{f}*}^\top (\mathbf{K}_{\mathbf{ff}} + \tilde{\mathbf{T}}^{-1})^{-1}\mathbf{k}_{\mathbf{f}*}) \\ &= \mathcal{N}(f_*|\, \mathbf{k}_{\mathbf{f}*}^\top (\tilde{\mathbf{T}}\mathbf{K}_{\mathbf{ff}} + \mathbf{I}_n)^{-1}\tilde{\boldsymbol{\nu}}, \\ &\qquad k_{**} - \mathbf{k}_{\mathbf{f}*}^\top \tilde{\mathbf{T}}^{1/2}(\tilde{\mathbf{T}}^{1/2}\mathbf{K}_{\mathbf{ff}}\tilde{\mathbf{T}}^{1/2} + \mathbf{I}_n)^{-1}\tilde{\mathbf{T}}^{1/2}\mathbf{k}_{\mathbf{f}*}) \ , \quad (5.27)\end{aligned}$$

where the last row expresses it in terms of the covariance matrix and site parameters only.

The predictive distribution at any new test point corresponds to integrating out the latent variable:

$$p(y_*|\mathbf{X},\mathbf{y},\mathbf{x}_*) = \int p(y_*|f_*)q(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*)\mathrm{d}f_* \quad (5.28)$$

This latter integral may or may not be tractable, depending on the form of the likelihood.

5.2.1.5 Summary of the procedure

The above procedure can summarized as follows:

1. Initialization:
   (a) Set all natural site parameters to zero $\{\tilde{\tau}_j\}_{j=1}^n$, $\{\tilde{\nu}_j\}_{j=1}^n$.
   (b) The corresponding approximate posterior parameters will be $\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{ff}}$ and $\boldsymbol{\mu} = \mathbf{0}$.
2. For each j, compute
   (a) The cavity parameters $\tau_{\backslash j}$ and $\nu_{\backslash j}$ using (5.19).
   (b) The moments of the tilted distribution, using equations derived from (5.21) and (5.22). (These equations will depend on the choice of the likelihood).
   (c) The new site parameters $\tilde{\tau}_j$ and $\tilde{\nu}_j$ using (5.23).
   (d) Update the approximate posterior parameters $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ using (5.15). Marginals $\{\sigma_j^2\}_{j=1}^n$ and $\{\mu_j\}_{j=1}^n$ are readily obtained from the diagonal of $\boldsymbol{\Sigma}$ and the elements of $\boldsymbol{\mu}$ respectively.
3. Repeat 2 until convergence.
4. Compute the NLML using (5.24).

Implementing EP in a fast and numerically stable way can be challenging in practice. There are several tricks and caveats that should be taken into account. We summarize the most relevant from Rasmussen and Williams (2006):

- In step 2.(d) it is not necessary to recompute $\boldsymbol{\Sigma}$ from scratch. The updated matrix can be expressed as a a rank-one update over the old one using (A.1). This means that this step only takes $\mathcal{O}(n^2)$ time.
- After a complete sweep over all sites, matrix $\boldsymbol{\Sigma}$ has received n rank-one updates, with total cost of $\mathcal{O}(n^3)$. To avoid loss of precision, it is recommended to recompute it form scratch after each complete sweep. This also takes $\mathcal{O}(n^3)$ time. Further numerical stability can be achieved by using the Cholesky decomposition to perform this step.

- The order in which the sites are visited in step 2 should be randomized for enhanced stability.
- Convergence of EP is not guaranteed for any arbitrary likelihood function. It has been conjectured that the algorithm always converge if likelihood is log-concave (which results in unimodality of the posterior).

One should note that the computational complexity of this algorithm is the same as that of a standard GP, $\mathcal{O}(n^3)$. The bottleneck is step 2.(d), which takes $\mathcal{O}(n^2)$ time, since the remaining steps within the loop are performed in $\mathcal{O}(1)$. The actual running time of this algorithm is of course much bigger in practice than that of a standard GP, since it involves more computations and several sweeps are needed to reach a fixed point.

5.2.2 EP for sparse GP models

The work of Quiñonero-Candela and Rasmussen (2005) suggests that most approximate GP models can be expressed as exact GP models in which the covariance function has been replaced by an approximation. The approximate covariance function results computationally advantageous because it yields a covariance matrix that can be expressed as $\mathbf{K_{ff}} = \mathbf{D}_0 + \mathbf{P}_0\mathbf{R}_0^\top\mathbf{R}_0\mathbf{P}_0^\top$, where $\mathbf{D}_0$ is an $n \times n$ diagonal matrix, $\mathbf{P}_0$ is an $n \times m$ matrix, and $\mathbf{R}_0$ is an $m \times m$ upper Cholesky factor. With this structure, the necessary operations (i.e. computing the NLML of the model, its derivatives and the predictive distributions) can be performed is $\mathcal{O}(m^2n)$ time, using matrix inversion lemma (A.1), instead of the original $\mathcal{O}(n^3)$ time. All the sparse GP models introduced in this thesis have the mentioned structure so we have been able to use (A.1) to produce computationally efficient algorithms. However, when dealing with non-Gaussian likelihoods, to perform the above mentioned operations we need to run EP to obtain an approximate posterior over $\mathbf{f}$, which in general is an $\mathcal{O}(n^3)$ operation.

Now, we are going to show how to reduce the computational cost of the EP algorithm to $\mathcal{O}(m^2n)$ when the covariance matrix $\mathbf{K_{ff}}$ has the cited structure, and provide expressions for the NLML and predictive variances that can also be computed in $\mathcal{O}(m^2n)$ time. This means that all models introduced so far can be used with non-Gaussian likelihoods without increasing their computational complexity (though, as mentioned before, the constant multiplying factor will be considerably increased).

5.2.2.1 Posterior updates

In order to be able to run EP in $\mathcal{O}(m^2 n)$, we just need to reduce the computational cost of step 2.(d) to $\mathcal{O}(m^2)$, so that the overall cost after updating the n sites is $\mathcal{O}(m^2 n)$.

Following Naish-Guzman and Holden (2008), instead of storing $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ directly, we define them as

$$\boldsymbol{\mu} = \mathbf{a} + \mathbf{P}\boldsymbol{\gamma} \quad \boldsymbol{\Sigma} = \mathbf{D} + \mathbf{P}\mathbf{R}^\top \mathbf{R}\mathbf{P}^\top,$$

and store only auxiliary elements $\mathbf{D}$ ($n \times n$ diagonal matrix), $\mathbf{P}$ ($n \times m$ matrix), $\mathbf{R}$ ($m \times m$ upper Cholesky factor), $\mathbf{a}$ ($n \times 1$ vector), and $\boldsymbol{\gamma}$ ($m \times 1$ vector).

Vectors $\mathbf{a}$ and $\boldsymbol{\gamma}$ are initialized to zero and the remaining matrices are initialized so that $\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{ff}}$, as step 1 of EP requires. This must be possible, since the structure of the covariance matrix of the sparse GP model was required to have this form.

Using this structure, marginal posteriors are no longer represented within $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$, but they can be obtained as follows

$$\sigma_j^2 = d_j + ||\mathbf{R}\mathbf{p}_j||^2 \quad \mu_j = a_j + \mathbf{p}_j^\top \boldsymbol{\gamma}, \tag{5.29}$$

which are $\mathcal{O}(m^2)$ and $\mathcal{O}(m)$ operations, respectively. Values $\mathbf{p}_j$, a_j, and d_j are the rows of $\mathbf{P}$ (in column vector form), the elements of $\mathbf{a}$, and the elements of the diagonal of $\mathbf{D}$, respectively.

The inverse approximate posterior covariance matrix is defined from (5.15) as $\boldsymbol{\Sigma}^{-1} = \mathbf{K}_{\mathbf{ff}}^{-1} + \tilde{\mathbf{T}}$, so that if site parameter $\tilde{\tau}_j$ changes by $\Delta\tilde{\tau}_j = \tilde{\tau}_{j\text{new}} - \tilde{\tau}_j$, the updated covariance matrix is related to the old one by

$$\boldsymbol{\Sigma}_{\text{new}}^{-1} = \boldsymbol{\Sigma}^{-1} + \Delta\tilde{\tau}_j \mathbf{e}\mathbf{e}^\top \tag{5.30}$$

where $\mathbf{e}$ is an $n \times 1$ vector with the j-th element set to 1 and the remaining elements set to zero. We first expand the right hand side of (5.30) in terms of the defining matrices, using matrix inversion lemma (A.1):

$$\boldsymbol{\Sigma}^{-1} + \Delta\tilde{\tau}_j \mathbf{e}\mathbf{e}^\top = \mathbf{D}^{-1} + \Delta\tilde{\tau}_j \mathbf{e}\mathbf{e}^\top - \mathbf{D}^{-1}\mathbf{P}\mathbf{R}^\top(\mathbf{R}\mathbf{P}^\top\mathbf{D}^{-1}\mathbf{P}\mathbf{R}^\top + \mathbf{I}_m)^{-1}\mathbf{R}\mathbf{P}^\top\mathbf{D}^{-1}.$$

Collecting $\mathbf{E} = \mathbf{D}^{-1} + \Delta\tilde{\tau}_j \mathbf{e}\mathbf{e}^\top$, which is also diagonal, and inverting the previous equation using matrix inversion lemma (A.1) once more,

$$\begin{aligned}
&\boldsymbol{\Sigma}_{\text{new}} = (\boldsymbol{\Sigma}^{-1} + \Delta\tilde{\tau}_j \mathbf{e}\mathbf{e}^\top)^{-1} = \\
&\mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{D}^{-1}\mathbf{P}\mathbf{R}^\top(\mathbf{R}\mathbf{P}^\top(\mathbf{D}\mathbf{E}\mathbf{D})^{-1}\mathbf{P}\mathbf{R}^\top - \mathbf{I}_m - \mathbf{R}\mathbf{P}^\top\mathbf{D}^{-1}\mathbf{P}\mathbf{R}^\top)^{-1}\mathbf{R}\mathbf{P}^\top\mathbf{D}^{-1}\mathbf{E}^{-1} \\
&= \mathbf{D}_{\text{new}} + \mathbf{P}_{\text{new}}\mathbf{R}_{\text{new}}^\top\mathbf{R}_{\text{new}}\mathbf{P}_{\text{new}}^\top,
\end{aligned} \tag{5.31}$$

so that the structure remains unchanged after the update. It is then possible to update $\boldsymbol{\Sigma}$ without ever computing it, but updating its defining matrices instead. Identifying terms in the last two rows of (5.31), we obtain the update equations:

$$\mathbf{D}_{\text{new}} = \mathbf{E}^{-1} = \mathbf{D} - \frac{\Delta\tilde{\tau}_j d_j^2}{1 + \Delta\tilde{\tau}_j d_j}\mathbf{e}\mathbf{e}^\top \tag{5.32a}$$

$$\mathbf{P}_{\text{new}} = \mathbf{E}^{-1}\mathbf{D}^{-1}\mathbf{P} = \mathbf{P} - \frac{\Delta\tilde{\tau}_j d_j}{1 + \Delta\tilde{\tau}_j d_j}\mathbf{e}\mathbf{p}_j^\top \tag{5.32b}$$

$$\begin{aligned}\mathbf{R}_{\text{new}} &= \text{chol}(\mathbf{R}^\top(\mathbf{R}\mathbf{P}^\top(\mathbf{D}\mathbf{E}\mathbf{D})^{-1}\mathbf{P}\mathbf{R}^\top - \mathbf{I}_m - \mathbf{R}\mathbf{P}^\top\mathbf{D}^{-1}\mathbf{P}\mathbf{R}^\top)^{-1}\mathbf{R}) \\ &= \text{chol}\left(\mathbf{R}^\top\left(\mathbf{I}_m - \mathbf{R}\mathbf{p}_j\frac{\Delta\tilde{\tau}_j}{1+\sigma_j^2}\mathbf{p}_j^\top\mathbf{R}^\top\right)\mathbf{R}\right)\end{aligned} \tag{5.32c}$$

Updates (5.32a) and (5.32b) can be computed in $\mathcal{O}(1)$ and $\mathcal{O}(m)$ time, respectively. Update (5.32c) is actually a Cholesky downdate, since $\mathbf{R}_{\text{new}}^\top\mathbf{R}_{\text{new}} = \mathbf{R}^\top\mathbf{R} - \mathbf{v}\mathbf{v}^\top$, where $\mathbf{v} = \mathbf{R}\mathbf{p}_j\sqrt{\Delta\tilde{\tau}_j/(1+\sigma_j^2)}$. To be precise, this is only a downdate if $\frac{\Delta\tilde{\tau}_j}{1+\sigma_j^2}$ is positive. Downdates, together with loss of precision, can result in a non-positive definite posterior covariance matrix. In the rare occasions in which this happens, the posterior parameters must be recomputed from scratch. Updates, which will occur whenever $\frac{\Delta\tilde{\tau}_j}{1+\sigma_j^2}$ is negative, are not problematic since they guarantee positive definiteness. Both updates and downdates of an $m \times m$ Cholesky factor can be computed in $\mathcal{O}(m^2)$.

The posterior mean is computed from the relation $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \tilde{\boldsymbol{\nu}}$. We know that the covariance matrix has been updated to $\boldsymbol{\Sigma}_{\text{new}}$. If there is also a change in site parameter $\Delta\tilde{\nu}_j = \tilde{\nu}_{j\text{new}} - \tilde{\nu}_j$, the new mean and covariance matrices are related to the old ones by $\boldsymbol{\Sigma}_{\text{new}}^{-1}\boldsymbol{\mu}_{\text{new}} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \Delta\tilde{\nu}_j\mathbf{e}$. Solving for $\boldsymbol{\mu}_{\text{new}}$ and using (5.30) to express $\boldsymbol{\Sigma}$ in terms of $\boldsymbol{\Sigma}_{\text{new}}$ and equating to the new expression for the mean, we have

$$\boldsymbol{\mu}_{\text{new}} = \boldsymbol{\Sigma}_{\text{new}}(\boldsymbol{\Sigma}_{\text{new}}^{-1} - \Delta\tilde{\tau}_j)\boldsymbol{\mu} + \boldsymbol{\Sigma}_{\text{new}}\Delta\tilde{\nu}_j\mathbf{e} = \mathbf{a}_{\text{new}} + \mathbf{P}_{\text{new}}\boldsymbol{\gamma}_{\text{new}},$$

so that for that equality to hold, the values of $\mathbf{a}_{\text{new}}$ and $\boldsymbol{\gamma}_{\text{new}}$ must be

$$\mathbf{a}_{\text{new}} = \mathbf{a} + \frac{(\Delta\tilde{\nu}_j + \Delta\tilde{\tau}_j a_j)d_j}{1 + \Delta\tilde{\tau}_j d_j}\mathbf{e} \tag{5.32d}$$

$$\boldsymbol{\gamma}_{\text{new}} = \boldsymbol{\gamma} + \frac{(\Delta\tilde{\nu}_j - \Delta\tilde{\tau}_j\mu_j)d_j}{1 + \Delta\tilde{\tau}_j d_j}\mathbf{R}_{\text{new}}^\top\mathbf{R}_{\text{new}}\mathbf{p}_j, \tag{5.32e}$$

with both updates involving $\mathcal{O}(1)$ and $\mathcal{O}(m^2)$ computing time, respectively. We have thus proved that for sparse GP models, it is possible to complete the main loop of EP in a total time of $\mathcal{O}(m^2 n)$, using $\mathcal{O}(mn)$ storage space.

As said in the previous section, after a sweep through all sites has been completed, it is recommendable to recompute the posterior parameters from scratch. This refresh can also be performed in $\mathcal{O}(m^2 n)$, so it does not increase the overall computational cost. The refresh equations follow.

Expanding $\boldsymbol{\Sigma}_{\text{new}}^{-1} = \mathbf{K}_{\mathbf{ff}}^{-1} + \tilde{\mathbf{T}}$ with $\boldsymbol{\Sigma}_{\text{new}} = \mathbf{D}_{\text{new}} + \mathbf{P}_{\text{new}}\mathbf{R}_{\text{new}}^{\top}\mathbf{R}_{\text{new}}\mathbf{P}_{\text{new}}$ and $\mathbf{K}_{\mathbf{ff}} = \mathbf{D}_0 + \mathbf{P}_0\mathbf{R}_0^{\top}\mathbf{R}_0\mathbf{P}_0^{\top}$ and identifying terms, we have the covariance refresh equations

$$\mathbf{D}_{\text{new}} = (\mathbf{I}_n + \mathbf{D}_0\tilde{\mathbf{T}})^{-1}\mathbf{D}_0 \tag{5.33a}$$

$$\mathbf{P}_{\text{new}} = (\mathbf{I}_n + \mathbf{D}_0\tilde{\mathbf{T}})^{-1}\mathbf{P}_0 \tag{5.33b}$$

$$\mathbf{R}_{\text{new}} = \text{ro180}\left(\text{chol}\left(\text{ro180}(\mathbf{I}_m + \mathbf{R}_0\mathbf{P}_0^{\top}\tilde{\mathbf{T}}(\mathbf{I}_n + \mathbf{D}_0\tilde{\mathbf{T}})^{-1}\mathbf{P}_0\mathbf{R}_0^{\top})\right)^{\top}\right)\backslash\mathbf{R}_0 \tag{5.33c}$$

where Cholesky factorization $\text{chol}(\cdot)$ and backslash ($\backslash$) operators are described in Section A.3 of Appendix A. Operator $\text{ro180}(\cdot)$ rotates a matrix $180°$, so that each of its elements moves from position (p, q) to position $(m - p + 1, m - q + 1)$. Refreshes (5.33a), (5.33b) and (5.33c) take $\mathcal{O}(n)$, $\mathcal{O}(mn)$ and $\mathcal{O}(m^2 n)$ time, respectively, as it is clear from the involved operations.

Expanding now $\boldsymbol{\Sigma}_{\text{new}}$ in $\boldsymbol{\mu}_{\text{new}} = \boldsymbol{\Sigma}_{\text{new}}\tilde{\boldsymbol{\nu}} = \mathbf{a}_{\text{new}} + \mathbf{P}_{\text{new}}\boldsymbol{\gamma}_{\text{new}}$ and identifying terms, we obtain the refresh equations for the mean

$$\mathbf{a}_{\text{new}} = \mathbf{D}_{\text{new}}\tilde{\boldsymbol{\nu}} \tag{5.33d}$$

$$\boldsymbol{\gamma}_{\text{new}} = \mathbf{R}_{\text{new}}^{\top}\mathbf{R}_{\text{new}}\mathbf{P}_{\text{new}}^{\top}\tilde{\boldsymbol{\nu}}\ , \tag{5.33e}$$

which are computable in $\mathcal{O}(n)$ and $\mathcal{O}(mn)$ time, respectively.

5.2.2.2 Model selection and inference

Expression for the NLML (5.24), needed for ML-II model selection, can be re-expressed in a computationally cheaper form applying the matrix and determinant inversion lemmas (A.1), (A.2). Defining $\mathbf{F} = \mathbf{I}_m + \mathbf{D}_0\tilde{\mathbf{T}}$ and $\mathbf{A} = \mathbf{I}_m + \mathbf{R}_0\mathbf{P}_0^{\top}\mathbf{F}^{-1}\tilde{\mathbf{T}}\mathbf{P}_0\mathbf{R}_0^{\top}$, the NLML is:

$$\begin{aligned} -\log q(\mathbf{y}|\mathbf{X}) = &-\frac{1}{2}\sum_{j=1}^{n}\log\tilde{Z}_i^2\tilde{\tau}_j(1 + \tilde{\tau}_j d_j)) + \frac{1}{2}\log|\mathbf{A}| + \frac{1}{2}\tilde{\boldsymbol{\nu}}^{\top}(\mathbf{F}\tilde{\mathbf{T}})^{-1}\tilde{\boldsymbol{\nu}} \\ &- \frac{1}{2}(\tilde{\boldsymbol{\nu}}^{\top}\mathbf{F}^{-1}\mathbf{P}_0)\mathbf{R}_0^{\top}\mathbf{A}^{-1}\mathbf{R}_0(\mathbf{P}_0^{\top}\mathbf{F}^{-1}\tilde{\boldsymbol{\nu}}) + \frac{n}{2}\log(2\pi) \end{aligned} \tag{5.34}$$

which takes $\mathcal{O}(m^2 n)$ time, as can be deduced from considering the required operations. Analytical derivatives of (5.34) wrt the hyperparameters can be usually computed in $\mathcal{O}(m^2 n)$ time too (this is the case for all the sparse GP models introduced in the previous chapters).

Given the special structure of the covariance function, we can conclude that it is possible to expand $\mathbf{k}_{\mathbf{f}*} = \mathbf{P}_0 \mathbf{R}_0^\top \mathbf{R}_0 \mathbf{p}_*$ and $k_{**} = d_* + \mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{R}_0 \mathbf{p}_*$, where $\mathbf{p}_*$ is a vector of size $m \times 1$ and d_* is some scalar. With this considerations, the predictive distribution (5.27) can be re-expressed as:

$$q(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(f_*|\mu_{\text{EP}*}, \sigma^2_{\text{EP}*}) \tag{5.35a}$$

$$\mu_{\text{EP}*} = \mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1} \mathbf{R}_0 \mathbf{P}_0^\top \mathbf{F}^{-1} \tilde{\boldsymbol{\nu}} \tag{5.35b}$$

$$\sigma^2_{\text{EP}*} = d_* + \mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1} \mathbf{R}_0 \mathbf{p}_* \,. \tag{5.35c}$$

which also takes $\mathcal{O}(m^2 n)$ time.

For enhanced numerical accuracy, it is recommended to code (5.34) and (5.35) using Cholesky factorizations, as described in Section D.5 of Appendix D. The expressions provided in the appendix are, though equivalent, far simpler than those of Naish-Guzman and Holden (2008). The implementation of the derivatives of (5.34) using Cholesky factorizations is provided in Section E.4 of Appendix E.

5.2.2.3 Summary of the procedure

Fast EP for sparse GP models can be summarized as follows:

1. Initialization:
   (a) Set all natural site parameters to zero $\{\tilde{\tau}_j\}_{j=1}^n$, $\{\tilde{\nu}_j\}_{j=1}^n$.
   (b) Initialize the auxiliary matrices from the prior variance $\mathbf{P} = \mathbf{P}_0$, $\mathbf{R} = \mathbf{R}_0$, $\mathbf{D} = \mathbf{D}_0$, (from the structure of the prior covariance matrix $\mathbf{K}_{\mathbf{ff}} = \mathbf{D}_0 + \mathbf{P}_0 \mathbf{R}_0^\top \mathbf{R}_0 \mathbf{P}_0^\top$).
   (c) Initialize the auxiliary vectors from the prior mean, which is zero: $\mathbf{a} = \mathbf{0}$ and $\boldsymbol{\gamma} = \mathbf{0}$.
2. For each j, compute
   (a) The cavity parameters $\tau_{\backslash j}$ and $\nu_{\backslash j}$ using (5.19).

(b) The moments of the tilted distribution, using equations derived from (5.21) and (5.22). (These equations will depend on the choice of the likelihood).

(c) The new site parameters $\tilde{\tau}_j$ and $\tilde{\nu}_j$ using (5.23).

(d) Update the auxiliary matrices and vectors of the posterior using (5.32). Compute the marginals $\{\sigma_j^2\}_{j=1}^n$ and $\{\mu_j\}_{j=1}^n$ using (5.29).

3. After every site has been updated, refresh the posterior using (5.33).

4. Repeat 2 and 3 until convergence.

5. Compute the NLML using (5.34), make predictions using (5.35).

5.3 Sparse Robust Regression

The regression models considered so far assumed that every observation (output) y_j was the sum of a latent function of the input $f(\mathbf{x}_j)$ plus Gaussian noise. This model may not be well suited for all applications: Not only can noise contain spikes that would be highly improbable under the Gaussian assumption, but also data corruption can lead to some observations not being even related to the input. Additionally, depending on how data was collected, some samples may belong to a different input-output mapping than the rest, thus violating the i.i.d. assumption. Observations that, due to these or other problems, have an abnormally large deviation from their expected value are called outliers. In this section we will show how to extend previous sparse GP models to handle the presence of outliers in data.

5.3.1 Sparse GP models with Laplace noise

For a Gaussian noise prior, 99.7% of all observations should lie within 3 standard deviations of the mean. Thus the posterior mean must bend and distort itself to accomodate outliers within that distance. Adding a few outliers to a data set can completely spoil the predictions of the model, rendering it useless. Regression models that show resistance to outliers, producing valid predictions in spite of their presence are called robust regression models.

According to the Bayesian framework, one should reflect his beliefs about data in the form of priors. If we think that large deviations from the mean are more probable than a Gaussian would suggest, we should express that in the form of a leptokurtic (i.e. fat-tailed) noise prior. Such a noise prior would result in a robust model. There are several ways to specify a leptokurtic noise prior. One option is to model noise as the combination of two Gaussian distributions with different widths, in which a narrow Gaussian accounts for the usual Gaussian noise and a wider one takes care of the outliers. Another option is to use a single leptokurtic distribution, such as the Student-t or Laplace distributions.

Using a Laplace prior for the noise has two interesting advantages: It is log-concave (and thus implies a log-concave likelihood, which is critical for EP to converge reliably, as mentioned at the end of Section 5.2.1), and its probability density decays at the minimum possible rate as we move away from the mean, i.e., it has maximally fat tails (within the family of log-concave distributions). Neither the Student-t or the two-Gaussian mixture are log-concave distributions.

For these reasons, the Laplace distribution will be our prior of choice for noise in robust models. Most of the details for inference and model selection on sparse GP models with non-Gaussian likelihoods were already provided in Section 5.2.2. The only additional details that are specific to Laplace noise are the expression of the likelihood

$$p(y_j|f_j) = \frac{1}{2\sigma_{\mathrm{L}}} \exp\left(-\frac{|y_j - f_j|}{\sigma_{\mathrm{L}}}\right) \tag{5.36}$$

(where σ_{L} is the width parameter of the distribution, related to the noise power), and the zeroth, first, and second non-central moments of the tilted distribution for the Laplace likelihood. These can be computed using (5.21) and (5.22). From Kuss (2006), the moments are

$$m_{0j} = \hat{m}_{0j}/\sigma_{\mathrm{L}} \ , \quad m_{1j} = \hat{m}_{1j}\sigma_{\mathrm{L}} + y_j \ , \quad m_{2j} = \hat{m}_{2j}\sigma_{\mathrm{L}}^2 + 2\hat{m}_{1j}\sigma_{\mathrm{L}}y_j + y_j^2 \ , \tag{5.37}$$

where $\{\hat{m}_{kj}\}_{j=1}^{n}$ are defined by

$$\hat{m}_{0j} = \frac{a_j + b_j}{4} \exp(\hat{\sigma}^2_{\backslash j}/2 - \hat{\mu}_{\backslash j}) \tag{5.38a}$$

$$\hat{m}_{1j} = \frac{1}{a_j + b_j}(a_j(\hat{\mu}_{\backslash j} - \hat{\sigma}^2_{\backslash j}) + b_j(\hat{\mu}_{\backslash j} + \hat{\sigma}^2_{\backslash j})) \tag{5.38b}$$

$$\begin{aligned}\hat{m}_{2j} = &\frac{a_j}{a_j + b_j}(\hat{\sigma}^2_{\backslash j} + (\hat{\mu}_{\backslash j} - \hat{\sigma}^2_{\backslash j})^2) + \frac{b_j}{a_j + b_j}(\hat{\sigma}^2_{\backslash j} + (\hat{\mu}_{\backslash j} + \hat{\sigma}^2_{\backslash j})^2) \\ &- \frac{2}{a_j + b_j} \exp\left(-\frac{1}{2}\left(\hat{\sigma}^2_{\backslash j} + \frac{\hat{\mu}^2_{\backslash j}}{\hat{\sigma}^2_{\backslash j}} - \log\frac{2}{\pi}\right) + \hat{\mu}_{\backslash j} + \frac{3}{2}\log\hat{\sigma}^2_{\backslash j}\right)\end{aligned} \tag{5.38c}$$

and finally, a_j, b_j, $\hat{\mu}_{\backslash j}$ and $\hat{\sigma}^2_{\backslash j}$ are all defined as a function of the cavity parameters and target y_j

$$\hat{\mu}_{\backslash j} = (\tau^{-1}_{\backslash j}\nu_{\backslash j} - y_j)/\sigma_{\mathrm{L}} \tag{5.39a}$$

$$\hat{\sigma}^2_{\backslash j} = \tau^{-1}_{\backslash j}/\sigma^2_{\mathrm{L}} \tag{5.39b}$$

$$a_j = \operatorname{erfc}\left(\frac{\hat{\sigma}^2_{\backslash j} - \hat{\mu}_{\backslash j}}{\sqrt{2\hat{\sigma}^2_{\backslash j}}}\right) \tag{5.39c}$$

$$b_j = \exp(2\hat{\mu}_{\backslash j}) \operatorname{erfc}\left(\frac{\hat{\sigma}^2_{\backslash j} + \hat{\mu}_{\backslash j}}{\sqrt{2\hat{\sigma}^2_{\backslash j}}}\right) , \tag{5.39d}$$

where $\operatorname{erfc}(\cdot) = 1 - \operatorname{erf}(\cdot)$ is the complementary error function. The error function $\operatorname{erf}(\cdot)$ is defined in (5.3). As noted by Kuss (2006), it is necessary to take special care when computing these expressions not to run into numerical problems.

Observations at test points y_* are modeled as $y_* = f_* + \varepsilon$, as usual, but in this case ε stands for white Laplace noise. Since the posterior distribution of f_* is given by (5.35) and the distribution of ε is (5.36), the predictive mean and variance are straightforward to obtain

$$q_{\mathrm{Rob}}(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(y_*|\mu_{\mathrm{Rob}*}, \sigma^2_{\mathrm{Rob}*}) \tag{5.40a}$$

$$\mu_{\mathrm{Rob}*} = \mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1}\mathbf{R}_0\mathbf{P}_0^\top \mathbf{F}^{-1}\tilde{\boldsymbol{\nu}} \tag{5.40b}$$

$$\sigma^2_{\mathrm{Rob}*} = 2\sigma^2_{\mathrm{L}} + d_* + \mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1}\mathbf{R}_0\mathbf{p}_* \tag{5.40c}$$

where we have utilized the fact that $\mathbb{V}[\varepsilon] = 2\sigma^2_{\mathrm{L}}$. These predictive equations, coupled with minimizing (5.34) for ML-II model selection, can be used to perform robust regression with any sparse GP model. The only difference among models is how each one specifies $\mathbf{D}_0$ $\mathbf{P}_0$ and $\mathbf{R}_0$ (and therefore d_* and $\mathbf{p}_*$). In the next section we provide details for the robust BN-MCN model.

5.3.2 Robust BN-MCN

We already have all the equations for robust regression with any sparse GP model. To use any MN model, we express its covariance matrix $\frac{\sigma_0^2}{m\sigma_\mathrm{p}^2}\boldsymbol{\Phi}_\mathbf{f}^\top\boldsymbol{\Phi}_\mathbf{f}$ in the required form

$$\mathbf{D}_0 = \mathbf{0}\ ,\quad \mathbf{P}_0 = \sqrt{\frac{\sigma_0^2}{m\sigma_\mathrm{p}^2}}\boldsymbol{\Phi}_\mathbf{f}\ ,\quad \mathbf{R}_0 = \mathbf{I}_m\ , \tag{5.41}$$

which for test values implies $d_* = 0$ and $\mathbf{p}_* = \sqrt{\frac{\sigma_0^2}{m\sigma_\mathrm{p}^2}}\boldsymbol{\phi}(\mathbf{x}_*)$.

To specifically use BN-MCN, we apply the definitions of Section 3.1.2, $\phi(\mathbf{u},\mathbf{x}) = \cos(\mathbf{u}^\top\tilde{\mathbf{x}})$ (where $\tilde{\mathbf{x}} = [1,\ \mathbf{x}^\top]^\top$ is the augmented input vector) and $\sigma_\mathrm{p}^2 = 1/2$. Recall the parametrization $\mathbf{u}_i = [\varphi_i,\ (\mathbf{L}^{-1}\boldsymbol{\omega}_i)^\top]^\top$.

The complete algorithm is identical to 3.2.3, but uses EP to obtain the posterior distributions:

1. Run robust MCN-fixed:
   - Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, and σ_L^2 to some sensible values. (We will use: One half of the ranges of the input dimensions, the variance of the outputs $\{y_j\}_{j=1}^n$ and $\sigma_0^2/4$, respectively).
   - Fix $\{\boldsymbol{\omega}_i\}_{i=1}^m$ to random values drawn from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0},\mathbf{I}_D)$ (to approximate the ARD SE covariance function). Initialize $\{\varphi_i\}_{i=1}^m$ from a uniform distribution in $[0, 2\pi)$.
   - Run EP for sparse models as described in Section 5.2.2.3, using (5.41) for the initialization and (5.37)-(5.39d) to compute the moments. Minimize (5.34), the NLML of the model, wrt to σ_0^2, σ_L^2, $\{\ell_d\}_{d=1}^D$, and $\{\varphi_i\}_{i=1}^m$.
2. Run robust BN-MCN:
   - Initialize $\{\ell_d\}_{d=1}^D$, σ_0^2, σ_L^2, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$ to the values they had after MCN-fixed converged.
   - Set σ_Lmin^2 to the value found for σ_L^2 after convergence of MCN-fixed. Initialize σ_L^2 slightly above σ_Lmin^2. (We will use $1.5\sigma_\mathrm{Lmin}^2$).
   - Run EP as above. Minimize (5.34), the NLML of the model, wrt to $\{\ell_d\}_{d=1}^D$, σ_0^2, σ_L^2, $\{\boldsymbol{\omega}_i\}_{i=1}^m$ and $\{\varphi_i\}_{i=1}^m$, with the constraint $\sigma_\mathrm{L}^2 > \sigma_\mathrm{Lmin}^2$.

Predictions are made using (5.40). Numerically stable equations using Cholesky decompositions to compute the NLML and its derivatives are provided in Section D.5 of Appendix D and Section E.4.2 of Appendix E, respectively.

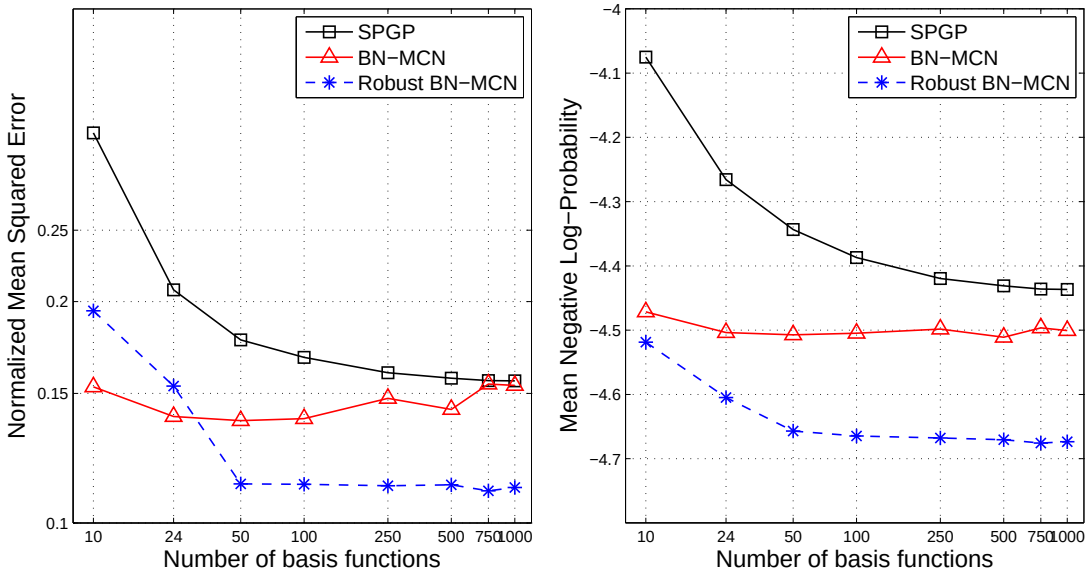
5.3.3 Experiments

This section has the modest aim of showing how the robust BN-MCN model performs, as compared to regular BN-MCN or regular SPGP. We therefore do not claim that robust BN-MCN produces better results than a possible robust version of SPGP, but that it produces better results than plain BN-MCN in the presence of outliers.

To introduce outliers in our data sets, we have randomly replaced 10% of the targets from the training set with random values. Those values are drawn from the distribution $\mathcal{N}(y|\frac{1}{n}\sum_{j=1}^{n} y_j, \frac{1}{n}\sum_{j=1}^{n} y_j^2 - (\frac{1}{n}\sum_{j=1}^{n} y_j)^2)$, so that they have the same mean and variance as the original set of targets. This, on the one hand, removes valuable data for inference, and on the other hand, adds misleading information, since for those data points there is no relation between inputs and outputs. Of course, the algorithms have no information about whether a data point is an outlier or not. The test data sets are left untouched. We use same performance measures (2.21) as in previous experiments.

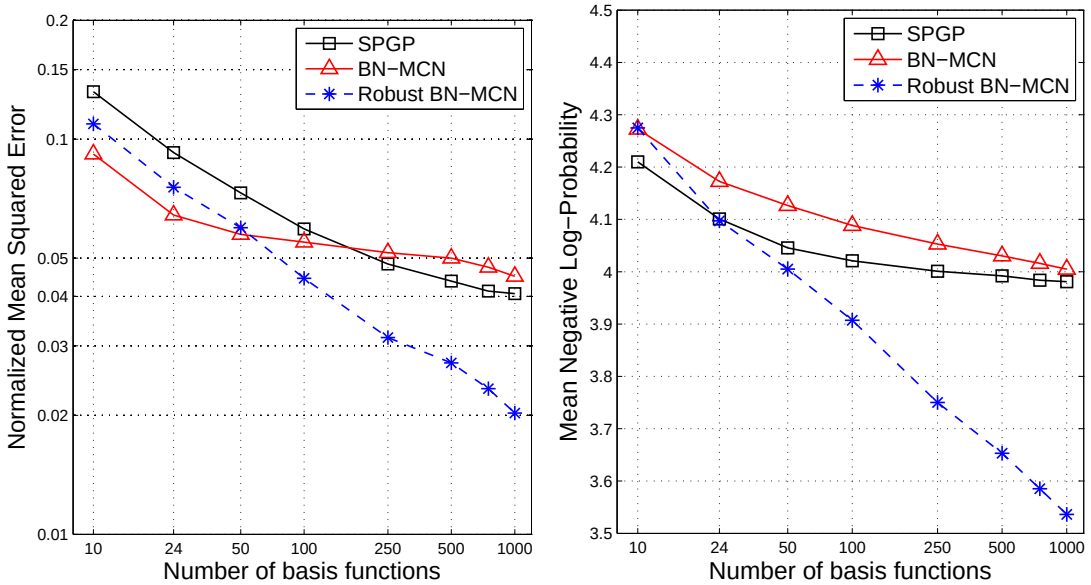
Figs. 5.11 and 5.12 show results for the *Elevators* and *Pole Telecomm* data sets. In both cases, the advantages of using a Laplace noise model over a Gaussian noise model are evident. In *Pole Telecomm* this difference becomes far more evident as the number of basis functions increases. Observe that neither the robust version of BN-MCN nor the Gaussian-noise version of SPGP show any apparent overfitting or overconfidence. Standard BN-MCN, on the other hand, is less robust against model misspecification and shows some overfitting.

In Figs. 5.13 and 5.14 we see the advantages of the robust method for the *Kin-40k* and *Pumadyn-32nm* data sets. For *Kin-40k*, the advantage is more noticeable when looking at the NMSE measure. The predictions made by robust BN-MCN on the *Pumadyn-32nm* data set are almost as good as those made with the original, non-corrupt data set. Robust BN-MCN not only correctly performs ARD, but it also seems to ignore the misleading information present in the outliers. In both data sets, SPGP lags behind and regular BN-MCN overfits.



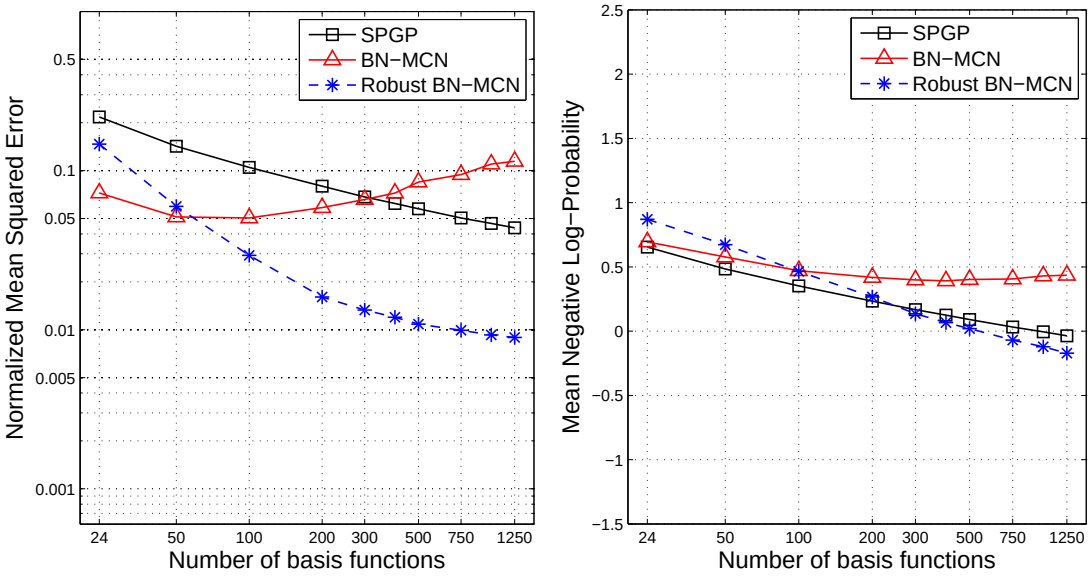
(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 5.11: NMSE and MNLP for SPGP, BN-MCN and Robust BN-MCN, for the *Elevators* problem.



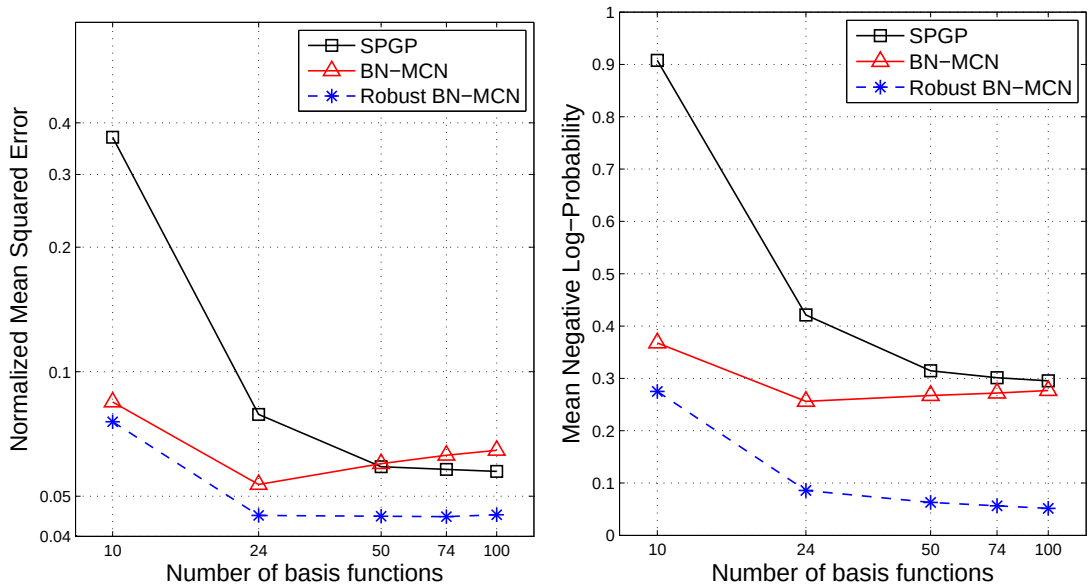
(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 5.12: NMSE and MNLP for SPGP, BN-MCN and Robust BN-MCN, for the *Pole Telecomm* problem.



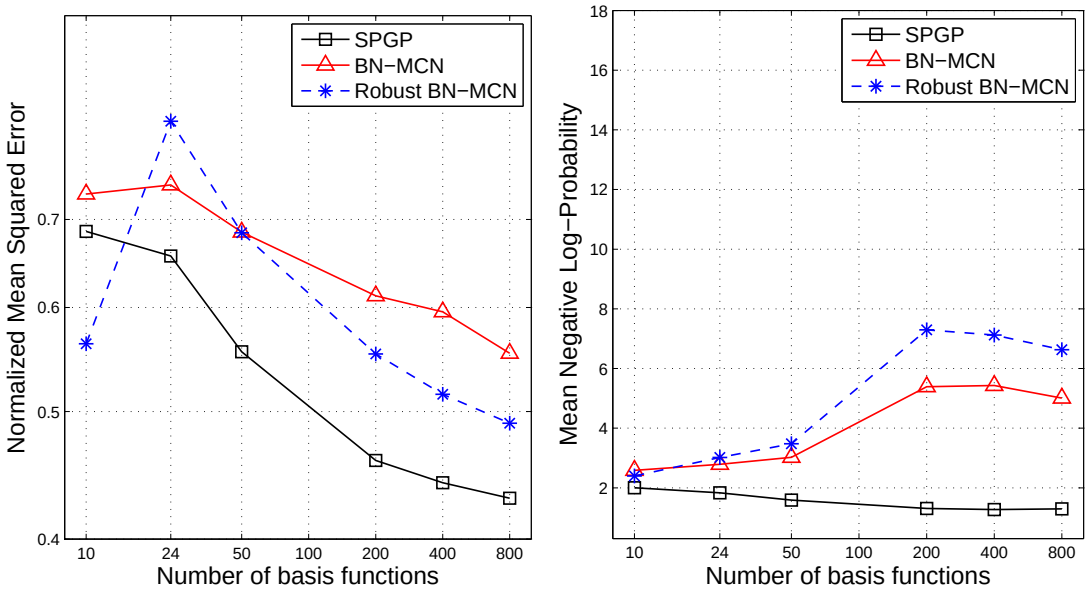
(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 5.13: NMSE and MNLP for SPGP, BN-MCN and Robust BN-MCN, for the *Kin-40k* problem.



(a) NMSE (log-log plot) (b) MNLP (semilog plot)

Figure 5.14: NMSE and MNLP for SPGP, BN-MCN and Robust BN-MCN, for the *Pumadyn-32nm* problem.



(a) NMSE (log-log plot)

(b) MNLP (semilog plot)

Figure 5.15: NMSE and MNLP for SPGP, BN-MCN and Robust BN-MCN, for the *Pendulum* problem.

Finally, results for data set *Pendulum* are shown in Fig. 5.15. The robust version of BN-MCN performs better than the regular one in terms on NMSE, but the clear winner for this data set is SPGP: As in previous data sets, it does not overfit despite model misspecification. For SPGP, inserting outliers in the data set only shows in the form of a performance hit. Both robust and regular versions of BN-MCN are unable to model this data set well and incur in low performance and overconfidence. If we compare the MNLP of regular BN-MCN when trained on a) the original data set (Fig. 3.6); and b) the data set corrupted with outliers (Fig. 5.15), it seems as if the inclusion of outliers resulted in a performance enhancement. Actually, performance is very bad in both cases, but inserting outliers increments the estimated noise level and therefore the uncertainty absolute minimum. This in turn results in better (but still too bad) MNLP.

5.3.4 Discussion

Two general conclusions can be drawn from these experiments:

(a) When dealing with corrupt data sets or noise of unknown nature, the use of robust

versions of the proposed regression methods is clearly advantageous. (Computation time increases considerably, though).

(b) The standard SPGP approximation is quite resistant against overfitting even in the presence of model misspecification.

5.4 Classification

Given a set of input-label pairs $\mathcal{D} \equiv \{\mathbf{x}_j, y_j\}_{j=1}^n$ with continuous inputs $\mathbf{x}_j \in \mathbb{R}^D$ and discrete labels $y_j \in \{-1, +1\}$, the binary classification task consists in assigning the appropriate label to some new, unseen test input $\mathbf{x}_*$. In the Bayesian setting, we are not only interested in knowing which of the two possible labels is more probable, but also how probable it is.

Approaches to classification can be either generative or discriminative. In the generative case we need to model class-conditional distribution $p(\mathbf{x}|y)$ and prior $p(y)$, from which posterior $p(y|\mathbf{x})$ is derived. In the discriminative approach the target probability distribution $p(y|\mathbf{x})$ is modeled directly, so that no assumptions about underlying distribution $p(\mathbf{x})$ need to be made. In the GP framework, the latter approach is preferred.

In this section we show how the sparse GP models described in this thesis can be extended to handle binary classification problems (with no increase in the complexity order).

5.4.1 GP classification

Here we provide a brief summary of GPs for classification, considering only the binary case. For a complete review, including the multi-class case, see Rasmussen and Williams (2006, Ch. 3).

A GP is an adequate prior to model arbitrary real-valued functions, so it can be directly applied to model the outputs of a regression data set. For classification tasks, however, we are interested in a model for $p(y = +1|\mathbf{x})$, a function which only takes values in the $[0, 1]$ range. It is possible to account for this by placing a GP prior over a latent, arbitrary-valued function $f(\mathbf{x})$ to which a sigmoid function $s(\cdot)$ is applied.

The sigmoid is required to fulfill $0 \leq s(z) \leq 1 \; \forall z \in \mathbb{R}$ and $s(-z) = 1 - s(z)$. The probability of a label given the corresponding input is then modeled as

$$p(y = +1|\mathbf{x}) = s(f(\mathbf{x})) \quad \text{with} \quad f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')), \tag{5.42}$$

where the GP is "squeezed" by the sigmoid so as to provide valid probability values. Note that the usual assumption of zero-mean GP $m(\mathbf{x}) = 0$ also makes sense in this context: By making the latent GP default back to zero at some $\mathbf{x}$, we make the corresponding probability default to $p(y|f(\mathbf{x}) = 0) = s(0) = 0.5$, so that both hypotheses are equiprobable (the latter identity follows from the symmetry requirement of the sigmoid).

Common choices for the sigmoid function are:

$$\text{logit}(z) = \frac{1}{1 + \mathrm{e}^{-z}} \tag{5.43}$$

$$\text{probit}(z) = \int_{-\infty}^{z} \mathcal{N}(t|0, 1)\mathrm{d}t = \frac{1}{2}\,\text{erf}\left(\frac{z}{\sqrt{2}}\right) + \frac{1}{2}, \tag{5.44}$$

where $\text{erf}(\cdot)$ is the standard error function defined by (5.3). Both sigmoid functions look quite similar (see Fig. 5.16) and can be used almost interchangeably, producing very similar results in practice.

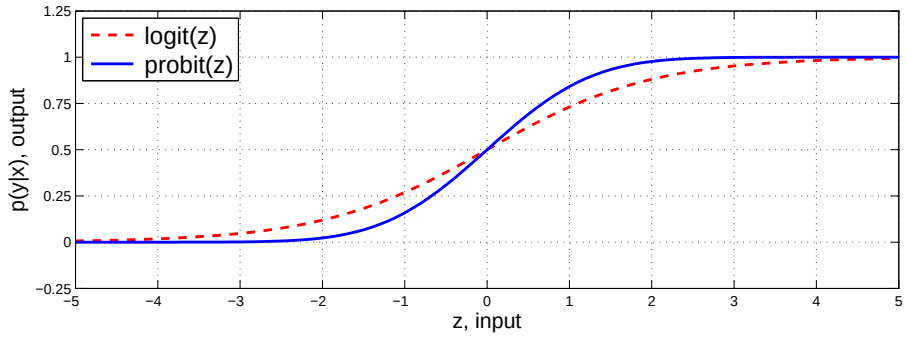


Figure 5.16: Examples of sigmoid functions for GP classification.

Since the probability of each label y_j only depends on the corresponding latent value $f_j = f(\mathbf{x}_j)$, it is possible to express the likelihood of the model as

$$p(\mathbf{y}|\mathbf{f}) = \prod_{j=1} p(y_j|f_j) \quad \text{with} \quad p(y_j|f_j) = s(y_j f_j) \tag{5.45}$$

where the latter identity follows from (5.42): When $y_j = +1$, $p(y_j|f_j) = s(f_j)$, and when $y_j = -1$, $p(y_j|f_j) = 1 - s(f_j) = s(-f_j)$. Both cases can be collected compactly as $p(y_j|f_j) = s(y_j f_j)$. Note that this is only possible due to the required sigmoid symmetry $1 - s(z) = s(-z)$.

Since (5.45) provides an expression for the (non-Gaussian) likelihood and the prior over $f(\mathbf{x})$ is a GP, it is possible to use the EP framework described in Section 5.2.1 for approximate model selection and inference. After computing the posterior distribution of the latent function at a new test point $\mathbf{x}_*$, the posterior probability of the labels at that point $p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ can be computed using (5.28).

The expressions for the moments of the tilted distribution, required for EP (see Section 5.2.1.3), as well as the (one-dimensional) integral (5.28) needed to compute the posterior probability, will depend of the chosen sigmoid function and may or may not be analytically tractable. In particular, when $s(z) = \text{probit}(z)$, the mentioned expressions are analytically tractable. In other cases, numerical integration may be needed.

Using $s(z) = \text{probit}(z)$, the moments of the tilted distribution are defined from the (natural) cavity parameters as

$$\mu_{\backslash j} = \tau_{\backslash j}^{-1} \nu_{\backslash j} \quad \sigma_{\backslash j}^2 = \tau_{\backslash j}^{-1} \tag{5.46}$$

$$m_{0j} = \text{probit}(z) \text{ with } z = \frac{y_j \mu_{\backslash j}}{\sqrt{1 + \sigma_{\backslash j}^2}} \tag{5.47}$$

$$m_{1j} = \mu_{\backslash j} + \frac{\sigma_{\backslash j}^2 \mathcal{N}(z|0,1)}{m_{0j} y_j \sqrt{1 + \sigma_{\backslash j}^2}} \tag{5.48}$$

$$m_{2j} = 2\mu_{\backslash j} m_{1j} - \mu_{\backslash j}^2 + \sigma_{\backslash j}^2 - \frac{z \sigma_{\backslash j}^4 \mathcal{N}(z|0,1)}{m_{0j}(1 + \sigma_{\backslash j}^2)} \tag{5.49}$$

and the posterior probability is determined from the posterior mean and variance of the latent function as

$$\begin{aligned} p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) &= \int p(y_*|f_*) q(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) \mathrm{d}f_* \\ &= \int \text{probit}(y_* f_*) \mathcal{N}(f_*|\mu_{\text{EP}*}, \sigma_{\text{EP}*}^2) \mathrm{d}f_* = \text{probit}\left(\frac{y_* \mu_{\text{EP}*}}{\sqrt{1 + \sigma_{\text{EP}*}^2}}\right). \end{aligned} \tag{5.50}$$

In principle, it could be interesting to adjust the scale of the $\text{probit}(z)$ function by introducing some hyperpameter h_w and using $\text{probit}(z/h_w)$ instead. However, as

explained in Rasmussen and Williams (2006, Ch. 3), adding a constant white noise term to the covariance function (i.e., a term $\sigma^2\delta_{\mathbf{x}\mathbf{x}'}$ where $\delta_{\mathbf{x}\mathbf{x}'}$ is a Kronecker delta) has the same scaling effect. The scale is then adjusted by learning σ^2.

5.4.2 Sparse GP classification

Using the framework from Section 5.2, it is straightforward to extend GP classification to sparse GP models. If the prior covariance matrix can be expressed as a low-rank matrix plus a diagonal matrix, the efficient version of EP in Section 5.2.2 for sparse GP models can be used to compute EP updates (5.32), NLML (5.34) and posterior distribution over the latent function (5.35) in $\mathcal{O}(m^2 n)$. As noted before, numerically stable versions of (5.34) and (5.35), described in Section D.5 of Appendix D, are preferred.

Assuming $s(z) = \text{probit}(z)$, posterior probability $p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ is obtained by plugging the posterior marginal (5.35) obtained after EP convergence into (5.50):

$$p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \text{probit}\left(\frac{y_*\mu_{\text{EP}*}}{\sqrt{1+\sigma^2_{\text{EP}*}}}\right) = \text{probit}\left(\frac{y_*\mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1}\mathbf{R}_0\mathbf{P}_0^\top \mathbf{F}^{-1}\tilde{\boldsymbol{\nu}}}{\sqrt{1+d_*+\mathbf{p}_*^\top \mathbf{R}_0^\top \mathbf{A}^{-1}\mathbf{R}_0\mathbf{p}_*}}\right). \tag{5.51}$$

More efficient computation of $\mu_{\text{EP}*}$ and $\sigma^2_{\text{EP}*}$ can be achieved using (D.5).

5.4.3 FIFGP for Classification (FIFGPC)

The description above applies to any sparse GP model with the specified covariance structure. We provide here the exact details for the FIFGP model, which will be used in the experimental section, but the procedure is analogous for any of the regression models presented so far.

The prior covariance matrix (including noise) of any IDGP is $\mathbf{K_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{K_{fu}}^\top + \boldsymbol{\Lambda}_\mathbf{y}$, where $\mathbf{K_{fu}}$ and $\mathbf{K_{uu}}$ are defined in terms of $k_{\text{INT}}(\cdot,\,\cdot)$ and $k_{\text{TR}}(\cdot,\,\cdot)$, as per (4.4). The covariance structure required in Section 5.2.2 is already present, so by direct identification we have:

$$\mathbf{D}_0 = \boldsymbol{\Lambda}_\mathbf{y}\ ,\quad \mathbf{P}_0 = \mathbf{K_{fu}}\ ,\quad \mathbf{R}_0 = \text{chol}(\mathbf{K_{uu}}^{-1})\ . \tag{5.52}$$

To avoid the explicit inversion of $\mathbf{K_{uu}}$ (the covariance matrix of the inducing features), it is also possible to use

$$\mathbf{R}_0 = \text{chol}(\mathbf{K}_{\mathbf{uu}}^{-1}) = \text{ro180}(\text{chol}(\text{ro180}(\mathbf{K_{uu}}))^\top \backslash \mathbf{I}_m),$$

where Cholesky factorization $\text{chol}(\cdot)$ and backslash ($\backslash$) operators are described in Section A.3 of Appendix A. Operator $\text{ro180}(\cdot)$ rotates a matrix $180°$, as described in Section 5.2.2.1.

For a test case, the corresponding diagonal element is $d_* = \sigma^2 + k_{**} - \mathbf{k}_{\mathbf{u}*}^\top \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{k}_{\mathbf{u}*}$ and the corresponding row of $\mathbf{P}_0$ is $\mathbf{p}_* = \mathbf{k}_{\mathbf{u}*}$.

These definitions are valid for any IDGP, including SPGP. To specifically use FIFGP, definitions (4.12) and (4.13) of Section and 4.3.3 must be used. These definitions approximate the ARD SE covariance function in the input domain. Since we will make comparisons with methods that use an isotropic Gaussian kernel, a single length-scale ℓ will be used for every input dimension, so that FIFGP approximates the isotropic SE covariance function. The complete algorithm, which we will call FIFGPC to note that it is a classification extension to FIFGP, is:

1. Initialize $\sigma_0^2 = 1$, $\sigma^2 = 1$, ℓ to the average of the ranges of all input dimensions and $\{c_d\}_{d=1}^D$ to the standard deviation of input data.

2. Initialize $\{\boldsymbol{\omega}\}_{i=1}^m$ from $\mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \ell^{-2}\mathbf{I}_D)$ and $\{\omega_{0i}\}_{i=1}^m$ from a uniform distribution in $[0, 2\pi)$.

3. Run EP for sparse models as described in Section 5.2.2.3, using (5.52) for the initialization and (5.49) to compute the moments. Minimize (5.34), the NLML of the model, wrt to ℓ, σ_0^2, σ^2, $\{c_d\}_{d=1}^D$ and $\{\boldsymbol{\omega}_i\}_{i=1}^m$. Since analytical derivatives are available, conjugate gradient descent can be used.

The probability that a new instance $\mathbf{x}_*$ belongs to class y_* can be computed using (5.51). Numerically stable equations using Cholesky decompositions to compute the NLML and its derivatives are provided in Section D.5 of Appendix D and Section E.4.1 of Appendix E, respectively.

5.4.4 Experiments

SPGP for classification (SPGPC) was introduced in Naish-Guzman and Holden (2008) and compared to several standard classification methods: Full GP Classification (GPC), Support Vector Machines (SVM) and Informative Vector Machines (IVM). The latter two are described, e.g., in Cortes and Vapnik (1995) and Lawrence et al. (2003), respectively. In this section we reproduce their results and add FIFGPC to the comparison.

For the experiments, the standard benchmark suite of Rätsch et al. (2001), consisting of 13 classification data sets, is used. Training sets sizes range between 400 and 1300, so that full GP classification is also feasible. All methods use the isotropic Gaussian as covariance function (also known as kernel in the SVM context). For each data set, 5 different random initializations on 10 different data splits are averaged. Hyperparameters are selected using ML-II, for which a maximum of 20 conjugate gradient iterations are allowed. For the SVM, hyperparameters are selected through cross-validation[3]. The error measures are:

$$\text{Erate} = \frac{\#\text{ of incorrectly classified samples}}{\#\text{ of test samples}}$$

$$\text{NLP} = \frac{1}{n_*}\sum_{j=1}^{n_*} \log p(y_{*j}|\mathbf{x}_{*j}, \mathcal{D}).$$

In previous experiments we have plotted the evolution of the error measures with m (the number of basis functions). In this case, since we are comparing so many data sets and methods, we will present the results in a more compact and comprehensive form: We follow Naish-Guzman and Holden (2008) and provide only the sparsest competitive solution for each method and data set. I.e., we select a value for m that, if reduced, would deteriorate performance, but if increased would not significantly improve it. For SVMs m is the number of support vectors, which is determined automatically.

Data sets, number of training and test instances and results for all methods are provided in Table 5.1. NLP is not provided for SVMs since they lack a direct probabilistic interpretation. Observe that FIFGPC, when compared with state-of-the-art classifiers

[3]This is the reason why an isotropic kernel is used in the experiments: It only needs 2 hyperparameters, as opposed to the ARD SE kernel, which requires $D + 1$. Given that cross-validation computation time scales exponentially with the number of hyperparameters, using the ARD SE kernel could make model selection unfeasible for some data sets.

such as SVMs or IVMs turns out to be quite competitive: It provides roughly the same performance whereas using only a fraction of the basis functions. SPGPC is even better: The number of basis functions required to solve some problems is astonishingly small: In 6 out the 13 data sets, state-of-the-art performance was achieved with only 2 basis functions! This seems to imply that in those data sets the boundary between classes was very simple, with data points corresponding to each class being well localized. Since SPGPC uses local basis functions, it yields very sparse solutions for this type of problems. FIFGPC, on the other hand, uses global basis functions and, for very simple problems, cannot compete with SPGPC in terms of sparsity. Notice, however, that if we turn to a more complex data set such as "splice", both SPGPC and FIFGPC require 200 basis functions and they both achieve the same performance.

5.4.5 Discussion

We can sum up the results of these experiments as follows:

(a) We have shown with a practical example how the introduced sparse GP models can be extended from regression to classification.

(b) At least one of these models (FIFGP) is able to beat state-of-the-art classification methods in terms of computational cost (smaller m), while roughly keeping up in performance.

(c) SPGPC already achieved this, and it seems doubtful whether this method can be consistently outperformed, since it already attains state-of-the-art performance with an extraordinarily low number of basis functions.

Data set			GPC		SVM		IVM			SPGPC			FIFGPC		
name	train:test	D	Erate	NLP	Erate	m	Erate	NLP	m	Erate	NLP	m	Erate	NLP	m
banana	400:4900	2	0.105	0.237	0.106	151	0.105	0.242	200	0.107	0.261	20	0.107	0.243	20
breast-cancer	200:77	9	0.288	0.558	0.277	122	0.307	0.691	120	0.281	0.557	2	0.294	0.579	4
diabetes	468:300	8	0.231	0.475	0.226	271	0.230	0.486	400	0.230	0.485	2	0.238	0.487	20
flare-solar	666:400	9	0.346	0.570	0.331	556	0.340	0.628	550	0.338	0.569	3	0.343	0.571	20
german	700:300	20	0.230	0.482	0.247	461	0.290	0.658	450	0.236	0.491	4	0.231	0.492	20
heart	170:100	13	0.178	0.423	0.166	92	0.203	0.455	120	0.172	0.414	2	0.182	0.443	20
image	1300:1010	18	0.027	0.078	0.040	462	0.028	0.082	400	0.031	0.087	200	0.049	0.130	200
ringnorm	400:7000	20	0.016	0.071	0.016	157	0.016	0.101	100	0.014	0.089	2	0.056	0.202	80
splice	1000:2175	60	0.115	0.281	0.102	698	0.225	0.403	700	0.126	0.306	200	0.124	0.304	200
thyroid	140:75	5	0.043	0.093	0.056	61	0.041	0.120	40	0.037	0.128	6	0.045	0.126	20
titanic	150:2051	3	0.221	0.514	0.223	118	0.242	0.578	100	0.231	0.520	2	0.228	0.519	2
twonorm	400:7000	20	0.031	0.085	0.027	220	0.031	0.085	300	0.026	0.086	2	0.027	0.082	10
waveform	400:4600	21	0.100	0.229	0.107	148	0.100	0.232	250	0.099	0.228	10	0.099	0.218	50

Table 5.1: Test error rate (Erate), Negative Log Probability (NLP) and number of basis (m) for the 13 classification data sets in the benchmark suite of Rätsch, using a full GP and several sparse methods, including the proposed FIFGPC.

5.5 Summary and conclusions

In this chapter we have explored a few possible extensions of the ideas presented in previous chapters. Though far from complete, this exploration shows how classical networks can be trained as MNs and should help anyone interested in extending sparse GP models to non-Gaussian likelihoods to get a detailed and complete picture of the process.

We have shown how MLPs can be expressed in the form of MNs (a type of sparse GP introduced in Chapter 3). Using this structure (enhanced with simple noise bounding or more sophisticated network mixing), it is possible to outperform state-of-the-art SPGP. Furthermore, the similarity of the results of this chapter with those obtained in Chapter 3 (which used cosine activation functions, very different from the sigmoids used here), suggests that the form of the activation functions is not very relevant.

We have given step-by-step details on how to extend GP models with low rank plus diagonal covariance matrices to handle non-Gaussian likelihoods. These steps are also described (in a less general manner) in Naish-Guzman and Holden (2008), but here we provide much simpler expressions for the computation of the posterior distributions.

We have developed a robust regression extension for sparse GP models and applied it to BN-SSGP. Experiments show clearly improved robustness to model misspecification and data outliers with respect to the non-robust version.

Finally, the extension of sparse GP models for classification was described and applied to FIFGP. Though competitive with SVMs and IVMs, FIFGP for classification hardly provides any advantage with respect to SPGP for classification, a method introduced in Naish-Guzman and Holden (2008) and which we expect will be increasingly adopted, given its high accuracy and low computational cost.

Chapter 6

Conclusions and further work

In this thesis we have developed and evaluated several sparse GP models along with different model selection strategies. These sparse GP models try to retain the advantages of full GPs (high accuracy, probabilistic predictions, no overfitting) while being computationally efficient, reducing computation time and storage space from $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ to $\mathcal{O}(m^2 n)$ and $\mathcal{O}(mn)$, respectively. Comparisons with the current state-of-the-art approximation, the Sparse Pseudo-inputs GP (SPGP), show that the novel methods provide significant improvement in practice and can therefore be useful to tackle large-scale problems. Regression models with Gaussian noise have been the main focus of this work, but we have shown in Chapter 5 how these models can be readily extended to perform robust regression, classification, etc.

In the following we will summarize the main contributions of this thesis; provide a big picture of the proposed algorithms, contrasting their specific advantages and disadvantages; and wrap up with a brief discussion about further possible extensions of this work.

6.1 Contributions

- **The Sparse Spectrum GP (SSGP).** In Chapter 2 we introduced SSGP, in which the spectral interpretation of GPs was exploited to achieve sparsity. SSGP has several properties that set it apart from other sparse models: It has a truly stationary covariance (whereas most sparse GPs only *approximate* stationarity), it has no location parameters (such as the pseudo-inputs of SPGP, or the active set

of other models), and it uses global, periodic, basis functions (as opposed to the localized bases of most sparse GPs). Three equivalent interpretations of SSGP were provided, and in particular we showed how SSGP corresponds to a generalized linear model using cosine bases in which the phases have been integrated out.

Two different strategies to select the spectral points were proposed:

(a) **Fixed spectral points:** If we draw them from some probability distribution $p(\mathbf{s}_r)$ and let them fixed, SSGP approximates a stationary full GP whose covariance function is (up to a scaling) the inverse Fourier transform of $p(\mathbf{s}_r)$. Convergence is achieved as the number of spectral points tends to infinity. Very few hyperparameters (the same as for the full GP) need to be selected, so model selection is fast and no overfitting appears. The downside of this option is its limited performance.

(b) **Selectable spectral points:** If we learn the spectral points (in addition to the hyperparameters) using ML-II, accuracy is greatly increased. However, the additional degrees of freedom introduced in the approximation imply a risk of overfitting if the number of spectral points is large. We have shown empirically that overfitting (in the sense explained in Section 2.5) is rarely a problem (probably due to the phase integration property of the model), but predictive variances can sometimes be very poor (i.e. models may be overconfident).

- **Marginalized Networks (MNs).** In Chapter 3 we introduced MNs along with two overfitting reduction strategies. MNs are essentially generalized linear models in which the output weights have been integrated out. The input weights can be obtained from some distribution and fixed, or learned together with signal and noise hyperparameters. Just as with SSGP, fixing the input weights avoids overfitting altogether, but more basis functions are needed to obtain high performance. When input weights are learned, only a moderate number of basis functions are needed, but overfitting problems may appear. In order to overcome them, two new methods were proposed: Noise bounding and network mixing.

  (a) **Noise bounding** is straightforward to apply: It first estimates the amount of noise present in data and then it lower bounds the noise hyperparameter with that estimate. We argued from a theoretic perspective how this

could help to avoid overfitting and confirmed it experimentally. Though this method seems to avoid overfitting, it still may produce poor predictive variances.

(b) **Network mixing** is a procedure akin to bagging, where diversity is introduced using different random initializations, instead of using different subsets of data. We have shown empirically that this procedure enhances both predictive mean and variances, and it yields very accurate predictions.

It is possible to cast SSGP as particular type of MN with additional constraints on the input weights. These constraints (that produce a phase integration effect) seem to be the cause of SSGP's built-in overfitting resistance. As shown in Chapter 2, SSGP provides good results even without resorting to noise bounding or network mixing. Nonetheless, if the number of spectral points is not small in comparison with the number of samples, combining SSGP with noise bounding is recommended to avoid possible overfitting.

We have also explicitly shown how the structure of MNs allows for low-cost linear dimensionality reduction. This can be achieved by:

(a) **Enforcing dimensionality reduction by design**, i.e., learning a linear projection of the input data in a space of prespecified, smaller dimension.

(b) **Discovering the intrinsic dimension of the learned function** after training by means of Singular Value Decomposition (SVD) of the input weights matrix, which only takes $\mathcal{O}(mD^2)$ time.

- **Inter-Domain GPs (IDGPs).** In Chapter 4 we extended GPs across domain boundaries, showing how it was possible to couple variables lying in different domains to make joint inference. This was exploited to extend SPGP, allowing the inducing variables to lie in a different domain than input data. We introduced the concept of "inducing features", which describes how each inducing variable summarizes information about the data set. Previously existing approximations, such as the SPGP approximation itself or the Sparse Multi-scale GP (SMGP) from Walder et al. (2008), can be interpreted as particular instances of this framework. Regarding an approximation as a type of IDGP can provide further insights about it. For the concrete case of SMGPs, some post-hoc variance adjustments that were needed in the original model arise on their own in

the IDGP framework and also additional constraints are shown to be necessary for the model to remain probabilistically well-defined.

IDGPs can be applied to any domain which linearly transforms the input domain (including convolutions, integrals and derivatives). In this work we developed

(a) IDGPs for (a "blurred" version of) the frequency domain in the form of a **Frequency Inducing-Features GP (FIFGP)**

(b) IDGPs for a mixed **Time-Frequency Inducing-Features GP (TFIFGP)**.

Both methods share the main properties of SPGP, but yield an overall higher performance on the tested data sets.

- Extension of MNs to the Multi-Layer Perceptron (MLP) case and general extension of sparse methods to non-Gaussian likelihoods. In Chapter 5 we do not introduce any new ideas, but provide concrete details and some experiments on how previous approaches can be extended. MNs are extended to the MLP case and previously seen sparse methods are adapted to handle non-Gaussian likelihoods, thus enabling them to perform robust regression and classification.

In summary, this thesis provides a set of new sparse GP models that can be used to tackle large-scale problems and compare favorably to the current state of the art. Flexibility and extendability have been emphasized throughout, so that tailoring these models to the specific needs of any concrete task is straightforward.

6.2 A comprehensive comparison of the new techniques

A comparison of the models discussed in this thesis is provided in Table 6.1. Each model class is marked according to three relevant characteristics: Accuracy (of mean predictions), overfitting resistance (in the sense described in Section 2.5), and quality of the predictive variances (i.e., whether they avoid overconfident predictions). As one could expect, none of the proposed models excels in every aspect. However, given the wide range of available options, we can expect at least one of them to be appropriate for any given task.

Method	Accuracy	Overfitting resistance	Uncertainty estimate quality
SSGP-fixed, MN-fixed (MCN-fixed, MMLP-fixed)	medium	very high	high
SSGP	high	medium	low-medium
BN-MNs (BN-MCN, BN-MMLP)	high	high	low
MNmix (MCNmix, MMLPmix)	very high	high	medium-high
IDGPS (SPGP, FIFGP, TFIFGP)	medium-high	high	high

Table 6.1: Comparative chart of the strengths and weaknesses of the methods introduced in this thesis. See the text for a description of each column.

It is also possible to further group these classes in two fundamentally different types of models:

(a) **Models which fit in the MN template** (i.e., have the structure of generalized linear models). When the input weights are learned, they achieve high accuracy but will probably overfit if used directly. Resorting to certain tricks, such as phase marginalization in SSGP, or the more general noise bounding and network mixing presented in Chapter 3, it is possible to reduce the impact of this problem while retaining the highly accurate predictions. Predictive variance quality may be not so good.

(b) **Models which fit in the IDGP template** (i.e., assume that all the values of the latent function are conditionally independent given a set of inducing variables). They have additional constraints to ensure that the target full GP is approximated as well as possible for the selected set of inducing features (by minimizing the Kullback-Leibler divergence, see Section 4.2). The proper handling of uncertainties in these models implies that high-quality predictive variances are provided.

However, being more constrained models, they are usually unable to achieve the high accuracy of MNs. They are also more complex models, so that their implementations will typically have longer computation times for the same number of basis functions.

6.3 Further work

Some of the relevant extensions to the main ideas of this thesis were already provided in Chapter 5. Many others, however, were left out and will be the subject of further research. To name a few:

- **Extensions to handle multiple outputs and multi-task learning.** The concept of "weight sharing" is used in NNs —among other things— to introduce some sort of coupling between different observations belonging to either the same input sample (multi-output regression) or different tasks (multi-task regression). This idea could not be used with standard GPs, since weights are integrated out, but can be applied to many of the models developed in this thesis (those that fit in the "MN template", first four rows of Table 6.1). This might be useful to build multi-output sparse GPs that benefit from the performance gains achieved by the discussed methods.

  Since multi-task learning deals with multiple data sets at once, its computational burden is particularly high. Bonilla et al. (2008) resort to the Nyström approximation to speed up computation, and also mention other alternatives such as SPGP. Some of our proposals, particularly IDGPs could also be well suited for this task, while providing a performance improvement. Furthermore, IDGPs would allow for a new type of coupling between tasks: "Inducing features sharing". By reusing the same inducing features over different tasks, it might be possible to learn them accurately with a smaller number samples per task, thus increasing predictive power.

- **New "feature extraction functions".** In this thesis we consider four different feature extraction functions, yielding SPGP, SMGP, FIFGP and TFIFGP. It would be interesting to explore other instances of this framework in the future; in particular it seems reasonable that using different windowing schemes (such

as multiple windows following the input data distribution) could result in better performing methods.

- **Combination with Titsias (2009) variational framework.** Another promising line of work would be to combine IDGPs with the variational method of Titsias (2009) to obtain sparse regression algorithms that more faithfully follow the full GP as the number of inducing features grows, eventually converging to it.

- **Applications needing cross-domain inference.** We mentioned this possibility in Chapter 4. Though we are not currently aware of applications where available data come from different, linearly related domains, such applications may arise in the future. Also, even when all data belongs to a single domain, this technique may be useful to make inference about features in another one (e.g., probabilistically inferring the amplitude of frequency components from data in time domain).

- **Combination of MNs with new regularization techniques.** One of the main problems that prevent the direct usage of MNs is the overfitting problem. Though we have provided specific workarounds in this thesis, it might be interesting to combine them with other (existing or future) regularization techniques.

Appendix A

Matrix algebra

The results stated in this appendix can be found in Golub and Loan (1989); Harville (1997); Lütkepohl (1996); Press et al. (2002).

A.1 Matrix inversion lemma

The matrix inversion lemma, also known as the Sherman–Morrison–Woodbury formula states that:

$$(\mathbf{Z} + \mathbf{U}\mathbf{W}\mathbf{V}^\top)^{-1} = \mathbf{Z}^{-1} - \mathbf{Z}^{-1}\mathbf{U}(\mathbf{W}^{-1} + \mathbf{V}^\top\mathbf{Z}^{-1}\mathbf{U})^{-1}\mathbf{V}^\top\mathbf{Z}^{-1} \tag{A.1}$$

When $\mathbf{Z}$ is an $n \times n$ diagonal matrix, $\mathbf{U}$ and $\mathbf{V}$ are $n \times m$ and $\mathbf{W}$ is $m \times m$, direct evaluation of the left hand side is $\mathcal{O}(n^3)$, whereas the right hand side can be cheaply evaluated in $\mathcal{O}(m^2 n)$.

A.2 Matrix determinant lemma

Analogously, the matrix determinant lemma states that:

$$|\mathbf{Z} + \mathbf{U}\mathbf{W}\mathbf{V}^\top| = |\mathbf{Z}||\mathbf{W}||\mathbf{W}^{-1} + \mathbf{V}^\top\mathbf{Z}^{-1}\mathbf{U}| \tag{A.2}$$

The same considerations as for matrix inversion lemma apply.

A.3 The Cholesky factorization

Any positive definite matrix can be expressed as a product of a lower triangular matrix and its transpose as follows:

$$\mathbf{A} = \mathbf{R}^\top \mathbf{R} \quad \rightarrow \quad \mathbf{R} = \mathrm{chol}(\mathbf{A}). \tag{A.3}$$

$\mathbf{R}$ is the upper Cholesky factor of $\mathbf{A}$. The lower Cholesky factor is defined as $\mathbf{L} = \mathbf{R}^\top$. If $\mathbf{A}$ is of size $n \times n$, the Cholesky factor can be computed in $\mathcal{O}(n^3/6)$ time and is a numerically very stable operation. This factorization is useful to solve linear systems, since

$$\mathbf{A}\mathbf{x} = \mathbf{b} \;\; \Leftrightarrow \;\; \mathbf{x} = \mathbf{R} \backslash (\mathbf{R}^\top \backslash \mathbf{b}), \tag{A.4}$$

where $\mathbf{C}\backslash\mathbf{c}$ denotes the solution to the linear system $\mathbf{C}\mathbf{x} = \mathbf{c}$. The solution of the two linear systems can be computed by forward and backward substitution in $\mathcal{O}(n^2/2)$ time each, rendering this method faster and more accurate than directly solving $\mathbf{A}\mathbf{x} = \mathbf{b}$.

As a by-product of the computation of $\mathbf{R}$, the determinant of $\mathbf{A}$ is readily available

$$|\mathbf{A}| = \prod_{i=1}^{n} \mathbf{R}_{ii}^2, \;\; \text{or equivalently} \;\; \frac{1}{2}\log|\mathbf{A}| = \sum_{i=1}^{n} \log \mathbf{R}_{ii}, \tag{A.5}$$

where $\mathbf{R}_{ii}$ denotes the i-th element of the diagonal of $\mathbf{R}$.

A.4 Matrix derivatives

The derivative of the elements of the inverse matrix $\mathbf{A}^{-1}$ is

$$\frac{\partial}{\partial\theta}\mathbf{A}^{-1} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial\theta}\mathbf{A}^{-1}. \tag{A.6}$$

If $\mathbf{A}$ is a positive definite symmetric matrix, the derivative of the log determinant is

$$\frac{\partial}{\partial\theta}\log|\mathbf{A}| = \mathrm{trace}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial\theta}\right). \tag{A.7}$$

Appendix B

Gaussian identities

The results stated in this Appendix can be found in most statistics textbooks, e.g. Mardia et al. (1979, ch. 3).

B.1 Multivariate Gaussian distribution

A random vector $\mathbf{x}$ is said to have a normal multivariate distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ if and only if

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv |2\pi\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]. \tag{B.1}$$

B.2 Marginal and conditional distributions

Given a joint Gaussian multivariate distribution

$$p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B}\end{bmatrix}\right),$$

the marginal distribution of $\mathbf{x}_1$ is

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1|\mu_1, \mathbf{A}), \tag{B.2}$$

and the distribution of $\mathbf{x}_1$ conditioned on $\mathbf{x}_2$ is

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\mu_1 + \mathbf{C}\mathbf{B}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top). \tag{B.3}$$

B.3 Integral of the product of two Gaussians

The following identity

$$\int \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})\mathcal{N}(\mathbf{a}|\mathbf{b}, \mathbf{B})d\mathbf{a} = \mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{A} + \mathbf{B}) \tag{B.4}$$

is useful to compute marginal likelihood $p(\mathbf{x})$ from likelihood $p(\mathbf{x}|\mathbf{a})$ and prior $p(\mathbf{a})$ when both are Gaussian.

B.4 Gaussian likelihood with linear parameter

If the mean of a Gaussian depends linearly on some parameter with Gaussian prior, it is possible to obtain the joint and posterior probabilities in closed form. Assume the following parametrization for the likelihood and the prior:

$$\begin{aligned} p(\mathbf{y}|\mathbf{w}, \mathbf{X}) &= \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{y}) \\ p(\mathbf{w}) &= \mathcal{N}(\mathbf{w}|\mathbf{a}, \boldsymbol{\Sigma}_\mathbf{w}) \ . \end{aligned}$$

The joint distribution is then $p(\mathbf{y}, \mathbf{w}|\mathbf{X}) = p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})$. This product can be conveniently arranged as:

$$p(\mathbf{y}, \mathbf{w}|\mathbf{X}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y} \\ \mathbf{w}\end{bmatrix} \middle| \begin{bmatrix}\mathbf{X}\mathbf{a} \\ \mathbf{a}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_\mathbf{y} + \mathbf{X}\boldsymbol{\Sigma}_\mathbf{w}\mathbf{X}^\top & \mathbf{X}\boldsymbol{\Sigma}_\mathbf{w} \\ \boldsymbol{\Sigma}_\mathbf{w}\mathbf{X}^\top & \boldsymbol{\Sigma}_\mathbf{w}\end{bmatrix}\right) \ . \tag{B.5}$$

The marginal likelihood of $\mathbf{y}$ is trivially contained in the previous expression due to the marginalization property of Gaussian distributions (B.2). The posterior distribution over $\mathbf{w}$ given $\mathbf{y}$ and $\mathbf{X}$ is obtained applying (B.3) to (B.5):

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_{\mathbf{w}|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{w}|\mathbf{y}}) \tag{B.6a}$$

$$\boldsymbol{\mu}_{\mathbf{w}|\mathbf{y}} = \mathbf{a} + \boldsymbol{\Sigma}_\mathbf{w}\mathbf{X}^\top(\boldsymbol{\Sigma}_\mathbf{y} + \mathbf{X}\boldsymbol{\Sigma}_\mathbf{w}\mathbf{X}^\top)^{-1}(\mathbf{y} - \mathbf{X}\mathbf{a}) \tag{B.6b}$$

$$\boldsymbol{\Sigma}_{\mathbf{w}|\mathbf{y}} = (\boldsymbol{\Sigma}_\mathbf{w}^{-1} + \mathbf{X}^\top\boldsymbol{\Sigma}_\mathbf{y}^{-1}\mathbf{X})^{-1} \ . \tag{B.6c}$$

B.5 Linear transformations

If $\mathbf{x}$ is a random vector following a multivariate Gaussian distribution

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{B.7}$$

then the linear transformation $\mathbf{y} = \mathbf{R}^\top \mathbf{x} + \mathbf{r}$ is also Gaussian distributed, as follows:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{R}^\top \boldsymbol{\mu} + \mathbf{r}, \mathbf{R}^\top \boldsymbol{\Sigma} \mathbf{R}). \tag{B.8}$$

B.6 Generation of random samples

Multivariate Gaussian samples of arbitrary covariance and mean can be obtained from univariate normal samples of mean 0 and variance 1. Just build a vector $\mathbf{x}$ of the desired dimension using independent univariate normal samples for each component. The desired sample $\mathbf{y}$ is then obtained as

$$\mathbf{y} = \mathbf{R}^\top \mathbf{x} + \boldsymbol{\mu} \tag{B.9}$$

where $\boldsymbol{\mu}$ is the desired mean and $\mathbf{R}$ is the upper Cholesky factor from the desired covariance matrix. This procedure follows trivially from (B.8).

Appendix C

Mathematical proofs

C.1 Rectangular-polar coordinate conversion

Here we prove the equivalence of two coordinate distributions when they are expressed in polar and rectangular form respectively. This equivalence is used to describe SSGP as a phase-integrating model in Section 2.1.2.2.

Let a and b be two independent random variables with known Gaussian probability densities

$$p(a) = \mathcal{N}(a|0, v) \quad \text{and} \quad p(b) = \mathcal{N}(b|0, v)$$

and $c \in \mathbb{R}$ and $\varphi \in [0, \pi]$ other two r.v. that satisfy the relation

$$\begin{cases} a = c\cos\varphi \\ b = c\sin\varphi \end{cases} \tag{C.1}$$

Pair $\{c, \varphi\}$ can be considered as the polar coordinates corresponding to rectangular coordinates $\{a, b\}$, with the particularity that c can take negative values and φ is restricted to one half of its usual range. Our task is to find the relation between the joint probability density of $\{c, \varphi\}$ and $\{a, b\}$:

$$p_{a,b}(a, b) = \frac{1}{2\pi v} \exp\left(-\frac{a^2 + b^2}{2v}\right)$$

Since the transformation of (C.1) provides a one-to-one mapping between them, the joint probability density of $\{c, \varphi\}$ is

$$p_{c,\varphi}(c, \varphi) = p_{a,b}(a, b)|J| = p_{a,b}(c\cos\varphi, c\sin\varphi)|J|,$$

where $|J|$ is the absolute value of Jacobian

$$J = \begin{vmatrix} \partial a/\partial c & \partial a/\partial \varphi \\ \partial b/\partial c & \partial b/\partial \varphi \end{vmatrix} = \begin{vmatrix} \cos\varphi & -c\sin\varphi \\ \sin\varphi & c\cos\varphi \end{vmatrix} = c$$

yielding

$$p_{c,\varphi}(c,\varphi) = \frac{1}{\pi}\frac{1}{2v}|c|\exp\left(-\frac{c^2}{2v}\right).$$

The marginals are

$$p_c(c) = \int_{\varphi=0}^{\pi} \frac{1}{\pi}\frac{1}{2v}|c|\exp\left(-\frac{c^2}{2v}\right)\mathrm{d}c = \frac{1}{2v}|c|\exp\left(-\frac{c^2}{2v}\right)$$

and

$$p_\varphi(\varphi) = \int_{c=-\infty}^{\infty} \frac{1}{2v}|c|\exp\left(-\frac{c^2}{2v}\right)\mathrm{d}\varphi = \frac{1}{\pi},$$

and, since $p_{c,\varphi}(c,\varphi) = p_c(c)p_\varphi(\varphi)$, we know that c and φ are independent.

The probability density of c is symmetric Rayleigh. Due to its symmetry, it is trivially zero mean, and its variance is

$$\mathbb{V}[c] = \int_{-\infty}^{\infty} c^2\frac{1}{2v}|c|\exp\left(-\frac{c^2}{2v}\right)\mathrm{d}c = 2\int_0^{\infty}\frac{1}{2v}c^3\exp\left(-\frac{c^2}{2v}\right)\mathrm{d}c = 2v$$

In this derivation, φ is uniform in $[0,\pi]$, but it is clear from the symmetry of (C.1) that any uniform density with range $[n_1\pi, n_2\pi]$, where $n_1, n_2 \in \mathbb{Z} : n_1 < n_2$, yields the same densities in a and b.

Given the one-to-one mapping between rectangular and polar coordinates, the converse must also be true; i.e., independent polar coordinates with the derived densities will result into the corresponding independent Gaussian densities when transformed back to rectangular.

C.2 Convergence of SSGP-fixed to a full GP for infinite bases

First recall the SSGP stochastic model (2.12), from Subsection 2.1.2.1, for the latent function:

$$f(\mathbf{x}) = \sum_{r=1}^{h}\left[a_r\cos(2\pi\mathbf{s}_r^\top\mathbf{x}) + b_r\sin(2\pi\mathbf{s}_r^\top\mathbf{x})\right],$$

where both a_r and b_r are random variables with densities

$$p(a_r) = \mathcal{N}(a_r|0, \sigma_0^2/h) \ , \quad p(b_r) = \mathcal{N}(b_r|0, \sigma_0^2/h) \ .$$

Since $f(\mathbf{x})$ is a linear combination of Gaussian-distributed random variables, it is a GP. The mean of this GP is zero:

$$\mathbb{E}[f(\mathbf{x})] = \sum_{r=1}^{h} \Big[\mathbb{E}[a_r]\cos(2\pi \mathbf{s}_r^\top \mathbf{x}) + \mathbb{E}[b_r]\sin(2\pi \mathbf{s}_r^\top \mathbf{x})\Big] = 0.$$

Its covariance is

$$\begin{aligned}
&\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] \\
&= \mathbb{E}\left[\sum_{r=1}^{h}\sum_{t=1}^{h}\Big[a_r\cos(2\pi \mathbf{s}_r^\top \mathbf{x}) + b_r\sin(2\pi \mathbf{s}_r^\top \mathbf{x})\Big]\Big[a_t\cos(2\pi \mathbf{s}_t^\top \mathbf{x}') + b_t\sin(2\pi \mathbf{s}_t^\top \mathbf{x}')\Big]\right] \\
&= \mathbb{E}\left[\sum_{r=1}^{h}\sum_{t=1}^{h} a_r a_t \cos(2\pi \mathbf{s}_r^\top \mathbf{x})\cos(2\pi \mathbf{s}_t^\top \mathbf{x}') + \sum_{r=1}^{h}\sum_{t=1}^{h} b_r b_t \sin(2\pi \mathbf{s}_r^\top \mathbf{x})\sin(2\pi \mathbf{s}_t^\top \mathbf{x}')\right] \\
&= \sum_{r=1}^{h}\left[\frac{\sigma_0^2}{h}\cos(2\pi \mathbf{s}_r^\top \mathbf{x})\cos(2\pi \mathbf{s}_r^\top \mathbf{x}') + \frac{\sigma_0^2}{h}\sin(2\pi \mathbf{s}_r^\top \mathbf{x})\sin(2\pi \mathbf{s}_r^\top \mathbf{x}')\right] \\
&= \frac{\sigma_0^2}{h}\sum_{r=1}^{h}\cos(2\pi \mathbf{s}_r^\top(\mathbf{x} - \mathbf{x}')),
\end{aligned}$$

which is the sample average of $\sigma_0^2\cos(2\pi \mathbf{s}_r^\top(\mathbf{x} - \mathbf{x}')$. If spectral samples $\{\mathbf{s}_r\}_{r=1}^h$ are distributed according to $p_S(\mathbf{s}) = \frac{S(\mathbf{s})}{\sigma_0^2}\, h$, when h tends to infinity, this average can be computed using an integral over the spectrum space. Calling $\boldsymbol{\tau} = \mathbf{x} - \mathbf{x}'$ and using (2.1):

$$\lim_{h\to\infty} \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \int_{\mathbb{R}^D} \sigma_0^2\cos(2\pi \mathbf{s}^\top \boldsymbol{\tau})\frac{S(\mathbf{s})}{\sigma_0^2}\mathrm{d}\mathbf{s} = \mathrm{Re}[k(\boldsymbol{\tau})] = k(\boldsymbol{\tau}).$$

Thus a full GP with any stationary covariance $k(\boldsymbol{\tau})$ can be recovered by using the SSGP model with infinite spectral points (equivalently, infinite basis functions) if they have the appropriate distribution, which can be derived from (2.2).

C.3 Convergence of MCN-fixed to a full GP for infinite bases

This demonstration is analogous to the previous one, but for the MCN stochastic model described in Section 3.1.2. Recall the model describing the latent function:

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{m} c_i \cos(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i)\;,$$

where $\{c_i\}_{i=1}^m$ are random variables with densities $p(c_i) = \mathcal{N}(c_i|0, 2\sigma_0^2/m)$.

Since $f(\mathbf{x})$ is a linear combination of Gaussian-distributed random variables, it is a GP. The mean of this GP is zero:

$$\mathbb{E}[f(\mathbf{x})] = \sum_{i=1}^{m} \mathbb{E}[c_i] \cos(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i) = 0.$$

Its covariance is:

$$\begin{aligned}
&\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \mathbb{E}\left[\sum_{i=1}^{m}\sum_{t=1}^{m} c_i c_t \cos(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i)\cos(2\pi \mathbf{s}_t^\top \mathbf{x}' - \varphi_t)\right] \\
&= \sum_{i=1}^{m} \frac{2\sigma_0^2}{m} \cos(2\pi \mathbf{s}_i^\top \mathbf{x} - \varphi_i)\cos(2\pi \mathbf{s}_i^\top \mathbf{x}' - \varphi_i) \\
&= \frac{2\sigma_0^2}{m}\sum_{i=1}^{m} \frac{1}{2}\left[\cos(2\pi \mathbf{s}_i^\top(\mathbf{x}+\mathbf{x}') - 2\varphi_i) + \cos(2\pi \mathbf{s}_i^\top(\mathbf{x}-\mathbf{x}'))\right] \\
&= \frac{\sigma_0^2}{m}\sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top(\mathbf{x}-\mathbf{x}')) + \frac{\sigma_0^2}{m}\sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top(\mathbf{x}+\mathbf{x}') - 2\varphi_i). \qquad \text{(C.2)}
\end{aligned}$$

The first term is identical to that found for the convergence of SSGP in Section C.2. It is the sample average of $\sigma_0^2 \cos(2\pi \mathbf{s}_i^\top(\mathbf{x}-\mathbf{x}')$. If spectral samples $\{\mathbf{s}_i\}_{i=1}^m$ are distributed according to $p_S(\mathbf{s}) = \frac{S(\mathbf{s})}{\sigma_0^2}$, when m tends to infinity, this average can be computed as an expectation over the spectrum space. Calling $\boldsymbol{\tau} = \mathbf{x} - \mathbf{x}'$ and using (2.1):

$$\lim_{m\to\infty} \frac{\sigma_0^2}{m}\sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top(\mathbf{x}-\mathbf{x}')) = \int_{\mathbb{R}^D} \sigma_0^2 \cos(2\pi \mathbf{s}^\top \boldsymbol{\tau}) \frac{S(\mathbf{s})}{\sigma_0^2} \mathrm{d}\mathbf{s} = \mathrm{Re}[k(\boldsymbol{\tau})] = k(\boldsymbol{\tau}).$$

The second term of covariance (C.2) can be interpreted as a perturbation to the SSGP covariance function due to explicitly considering phases (i.e., not integrating them out). However, when m tends to infinity, this perturbation vanishes, provided

that phases $\{\varphi_j\}_{j=1}^m$ are uniformly distributed in any $[n_1\pi, n_2\pi]$ range with $n_1, n_2 \in \mathbb{Z} : n_2 > n_1$. As before, in the infinite limit, the average can be replaced by an expectation:

$$\begin{aligned}
&\lim_{m\to\infty} \frac{\sigma_0^2}{m} \sum_{i=1}^{m} \cos(2\pi \mathbf{s}_i^\top (\mathbf{x} + \mathbf{x}') - 2\varphi_i) \\
&= \int_{n_1\pi}^{n_2\pi} \sigma_0^2 \cos(2\pi \mathbf{s}^\top (\mathbf{x} + \mathbf{x}') - 2\varphi) \frac{1}{(n_2 - n_1)\pi} \mathrm{d}\varphi \\
&= \frac{-\sigma_0^2}{2\pi(n_2 - n_1)} (\sin(2\pi \mathbf{s}^\top (\mathbf{x} + \mathbf{x}') - 2\pi n_2) - \sin(2\pi \mathbf{s}^\top (\mathbf{x} + \mathbf{x}') - 2\pi n_1)) = 0.
\end{aligned}$$

Therefore

$$\lim_{m\to\infty} \mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] = k(\boldsymbol{\tau}) \ .$$

Hence a full GP with any stationary covariance $k(\boldsymbol{\tau})$ can be recovered by using the MCN model with infinite basis functions if their frequencies and phases have the appropriate distribution, which can be derived from (2.2).

Appendix D

Model implementation

The algorithms developed in this thesis can be implemented in a faster and numerically wiser way using Cholesky factorizations, detailed in Section A.3. These implementations are derived applying (A.3), (A.4) and (A.5) to the original descriptions. $\mathbf{R}_{ii}$ denotes the i-th element of the diagonal of $\mathbf{R}$. Recall that $\mathbf{C}\backslash\mathbf{c}$ denotes the solution to the linear system $\mathbf{C}\mathbf{x} = \mathbf{c}$.

D.1 Full GP

Eqs. (1.15) and (1.18) can be implemented as:

$$\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I}_n) \tag{D.1a}$$

$$\boldsymbol{\alpha} = \mathbf{R}\backslash(\mathbf{R}^\top\backslash\mathbf{y}) \tag{D.1b}$$

$$\mathbf{v} = \mathbf{R}^\top\backslash\mathbf{k}_{\mathbf{f}*} \tag{D.1c}$$

$$\mu_* = \mathbf{k}_{*\mathbf{f}}\boldsymbol{\alpha} \tag{D.1d}$$

$$\sigma_*^2 = k_{**} - \mathbf{v}^\top\mathbf{v} \tag{D.1e}$$

$$-\log p(\mathbf{y}|\mathbf{X}) = \frac{1}{2}\mathbf{y}^\top\boldsymbol{\alpha} + \frac{1}{2}\sum_{i=1}^{n}\mathbf{R}_{ii} - \frac{n}{2}\log(2\pi)\ . \tag{D.1f}$$

D.2 Sparse Spectrum GP

Equations (2.10) and (2.11) can be implemented as:

$$\mathbf{R} = \text{chol}\left(\mathbf{\Phi_f}\mathbf{\Phi_f}^\top + \frac{h\sigma^2}{\sigma_0^2}\mathbf{I}_{2h}\right) \tag{D.2a}$$

$$\mu_* = \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{R}\backslash(\mathbf{R}^\top\backslash(\mathbf{\Phi_f}\mathbf{y})) \tag{D.2b}$$

$$\sigma_*^2 = \sigma^2 + \sigma^2||\mathbf{R}^\top\backslash\boldsymbol{\phi}(\mathbf{x}_*)||^2 \tag{D.2c}$$

$$\begin{aligned}-\log p(\mathbf{y}|\mathbf{X}) = &\frac{1}{2\sigma^2}\left[||\mathbf{y}||^2 - ||\mathbf{R}^\top\backslash(\mathbf{\Phi_f}\mathbf{y})||^2\right] \\ &+ \frac{1}{2}\sum_{i=1}^{m}\log\mathbf{R}_{ii}^2 - h\log\frac{h\sigma^2}{\sigma_0^2} + \frac{n}{2}\log 2\pi\sigma^2 \ .\end{aligned} \tag{D.2d}$$

Remember that h is used to name the number of spectral points; there are $m = 2h$ basis functions.

D.3 Marginalized Networks (also Bounded Noise and Network Mixture cases)

The implementation of (3.4) and (3.5) is as follows:

$$\mathbf{R} = \text{chol}\left(\mathbf{\Phi_f}\mathbf{\Phi_f}^\top + \frac{m\sigma^2\sigma_\text{p}^2}{\sigma_0^2}\mathbf{I}_m\right) \tag{D.3a}$$

$$\mu_* = \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{R}\backslash(\mathbf{R}^\top\backslash(\mathbf{\Phi_f}\mathbf{y})) \tag{D.3b}$$

$$\sigma_*^2 = \sigma^2 + \sigma^2||\mathbf{R}^\top\backslash\boldsymbol{\phi}(\mathbf{x}_*)||^2 \tag{D.3c}$$

$$\begin{aligned}-\log p(\mathbf{y}|\mathbf{X}) = &\frac{1}{2\sigma^2}\left[||\mathbf{y}||^2 - ||\mathbf{R}^\top\backslash(\mathbf{\Phi_f}\mathbf{y})||^2\right] \\ &+ \frac{1}{2}\sum_{i=1}^{m}\log\mathbf{R}_{ii}^2 - \frac{m}{2}\log\frac{m\sigma^2\sigma_\text{p}^2}{\sigma_0^2} + \frac{n}{2}\log 2\pi\sigma^2 \ .\end{aligned} \tag{D.3d}$$

D.4 Inter-Domain GP

The following implements (4.8) and (4.9):

$$\mathbf{R}_{\mathbf{uu}} = \text{chol}(\mathbf{K}_{\mathbf{uu}} + \epsilon \mathbf{I}_m) \tag{D.4a}$$
$$\mathbf{\Lambda} = \sigma^{-2}\mathbf{\Lambda}_{\mathbf{y}} \tag{D.4b}$$
$$\mathbf{\Phi}_{\text{s}} = (\mathbf{R}_{\mathbf{uu}}^{\top} \backslash \mathbf{K}_{\mathbf{fu}}^{\top})\mathbf{\Lambda}^{-1/2} \tag{D.4c}$$
$$\mathbf{R} = \text{chol}(\mathbf{\Phi}_{\text{s}}\mathbf{\Phi}_{\text{s}}{}^{\top} + \sigma^2 \mathbf{I}_m) \tag{D.4d}$$
$$\mathbf{y}_{\text{s}} = \mathbf{\Lambda}^{-1/2}\mathbf{y} \tag{D.4e}$$
$$\mathbf{H} = \mathbf{R}^{\top} \backslash \mathbf{\Phi}_{\text{s}} \tag{D.4f}$$
$$\boldsymbol{\beta} = \mathbf{H}\mathbf{y}_{\text{s}} \tag{D.4g}$$
$$\mathbf{v}_1 = \mathbf{R}_{\mathbf{uu}}^{\top} \backslash \mathbf{k}_{\mathbf{u}*} \tag{D.4h}$$
$$\mathbf{v}_2 = \mathbf{R}^{\top} \backslash \mathbf{v}_1 \tag{D.4i}$$
$$\mu_{\text{IDGP}*} = \boldsymbol{\beta}^{\top}\mathbf{v}_2 \tag{D.4j}$$
$$\sigma^2_{\text{IDGP}*} = \sigma^2 + k_{**} - ||\mathbf{v}_1||^2 + \sigma^2||\mathbf{v}_2||^2 \tag{D.4k}$$
$$\begin{aligned} -\log p(\mathbf{y}|\mathbf{X}) = {} & \frac{1}{2\sigma^2}(||\mathbf{y}_{\text{s}}^2|| - ||\boldsymbol{\beta}||^2) + \frac{1}{2}\sum_{i=1}^{m} \log[\mathbf{R}]_{ii}^2 \\ & + \frac{n-m}{2}\log\sigma^2 + \frac{1}{2}\sum_{j=1}^{n} \log 2\pi[\mathbf{\Lambda}]_{jj} \ , \end{aligned} \tag{D.4l}$$

where ϵ is some jitter noise to enhance numerical condition, typically 10^{-6}.

D.5 EP for sparse models

The main loop of EP for sparse models can be implemented as summarized in Subsection 5.2.2.3, where efficient and numerically stable update equations are provided. Here, we provide the corresponding expressions to compute the Negative Log-Marginal Likelihood (NLML) of the model and make inference at new test points.

After EP has converged, we have access to site parameters $\{\tilde{\nu}_j, \tilde{\tau}_j\}_{j=1}^n$, cavity parameters $\{\nu_{\backslash j}, \tau_{\backslash j}\}_{j=1}^n$, moments of the tilted distribution $\{m_{0j}, m_{1j}, m_{2j}\}_{j=1}^n$, and defining vectors and matrices for the posterior mean and covariance matrix $\mathbf{a}_{\text{new}}$, $\boldsymbol{\gamma}_{\text{new}}$, $\mathbf{D}_{\text{new}}$, $\mathbf{P}_{\text{new}}$, $\mathbf{R}_{\text{new}}$ (see Subsections 5.2.1 and 5.2.2). Also recall from Subsection 5.2.2.1 that the prior covariance matrix can be expressed as $\mathbf{K}_{\mathbf{ff}} = \mathbf{D}_0 + \mathbf{P}_0\mathbf{R}_0^{\top}\mathbf{R}_0\mathbf{P}_0^{\top}$

(where $\mathbf{D}_0$ is a diagonal $n \times n$ matrix with elements $\{d_j\}_{j=1}^n$, $\mathbf{P}_0$ is an $n \times m$ matrix, and $\mathbf{R}_0$ is an $m \times m$ upper Cholesky factor) and for any test point, it is possible to expand $\mathbf{k}_{\mathbf{f}*} = \mathbf{P}_0\mathbf{R}_0^\top\mathbf{R}_0\mathbf{p}_*$ and $k_{**} = d_* + \mathbf{p}_*^\top\mathbf{R}_0^\top\mathbf{R}_0\mathbf{p}_*$, where $\mathbf{p}_*$ is a vector of size $m \times 1$ and d_* is some scalar.

Using these values and some algebra, NLML (5.34) and the posterior distribution at some test point (5.35) can be expressed in a numerically safer and more stable way:

$$\mu_{\text{EP}*} = \mathbf{p}_*^\top \boldsymbol{\gamma}_{\text{new}} \tag{D.5a}$$

$$\sigma^2_{\text{EP}*} = d_* + ||\mathbf{R}_{\text{new}}\mathbf{p}_*||^2 \tag{D.5b}$$

$$\begin{aligned} -\log q(\mathbf{y}|\mathbf{X}) = &-\frac{1}{2}\sum_{i=1}^m \log[\mathbf{R}_{\text{new}}]_{ii}^2 + \frac{1}{2}\sum_{i=1}^m \log[\mathbf{R}_0]_{ii}^2 + \frac{1}{2}\sum_{j=1}^n \log(1+\tilde{\tau}_j d_j) \\ &- \frac{1}{2}\sum_{j=1}^n \log\left(1 + \frac{\tilde{\tau}_j}{\tau_{\setminus j}}\right) - \frac{1}{2}\sum_{j=1}^n \log(\tilde{\nu}_j[\boldsymbol{\mu}_{\text{new}}]_j) - \frac{1}{2}\sum_{j=1}^n \log m_{0j}^2 \\ &- \frac{1}{2}\sum_{j=1}^n \frac{\nu_{\setminus j}(\tilde{\tau}_j\tau_{\setminus j} - 2\tilde{\nu}_j) - \tilde{\nu}_j^2}{\tilde{\tau}_j + \tau_{\setminus j}} \end{aligned} \tag{D.5c}$$

Appendix E

Model log-evidence derivatives

This appendix provides, without proof, the derivatives of the Negative Log-Marginal Likelihood (NLML) of the models developed throughout this thesis. The NLML of each model is given in the text as a function of the hyperparameters, so obtaining these expressions is a mechanical (yet tedious) task. We use $\partial \mathbf{A}/\partial\theta$ to denote element-wise derivation of matrix $\mathbf{A}$ wrt scalar θ.

E.1 Length-scale and power hyperparameters

Unconstrained optimization algorithms (such as conjugate gradient descent) need the objective function to be defined in terms of arbitrary-valued hyperparameters. However, length-scales $\{\ell_d\}_{d=1}^D$ and power hyperparameters $\{\sigma_0^2, \sigma^2\}$ only accept positive values. Furthermore, σ^2 is sometimes constrained to be above some known value σ^2_{min}. In order to apply unconstrained optimization to our models, we transform the hyperparameters as follows

$$\theta_{\ell_d} = \log(\ell_d) \;, \quad \theta_{\sigma_0^2} = \frac{1}{2}\log(\sigma_0^2) \;, \quad \theta_{\sigma^2} = \frac{1}{2}\log(\sigma^2 - \sigma^2_{\text{min}}) \;,$$

and optimize over $\{\theta_{\ell_d}\}_{d=1}^D$, $\theta_{\sigma_0^2}$ and θ_{σ^2} instead.

Therefore, in the following sections, we will provide (among others) derivatives wrt these auxiliary hyperparameters so that they can be used with any gradient-based optimization procedure.

E.2 MNs log-evidence derivatives

Here we show how to compute the NLML derivatives for MNs, which are introduced in Chapter 3. Assume $\mathbf{R}$ defined as in (D.3). Then compute:

$$\mathbf{H} = (\mathbf{R}^\top \backslash \mathbf{\Phi_f})/\sigma \; ; \qquad \boldsymbol{\beta} = \mathbf{H}\mathbf{y} \; ; \qquad \mathbf{b} = \mathbf{y}/\sigma^2 - (\mathbf{H}^\top \boldsymbol{\beta})/\sigma \tag{E.1}$$

$$\mathbf{A} = [\mathbf{H}^\top, \; \mathbf{b}]^\top \; ; \qquad \mathbf{D_A} = \operatorname{diag}(\mathbf{A}^\top \mathbf{A}) \; ; \qquad \mathbf{D_\Phi} = \operatorname{diag}(\mathbf{\Phi_f}^\top \mathbf{\Phi_f}) \tag{E.2}$$

$$\mathbf{F}_\mathbf{\Phi}^\top = (\mathbf{\Phi_f}\mathbf{A}^\top)\mathbf{A} - \mathbf{\Phi_f}/\sigma^2 \; . \tag{E.3}$$

After these $\mathcal{O}(m^2 n)$ precomputations, derivatives wrt power hyperparameters (as defined in Section E.1) can be obtained from

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma_0^2}} = \frac{\sigma_0^2}{m\sigma_\mathrm{p}^2}(\operatorname{trace}(\mathbf{D_\Phi})/\sigma^2 - \operatorname{trace}((\mathbf{\Phi_f}\mathbf{A}^\top)(\mathbf{\Phi_f}\mathbf{A}^\top)^\top)) \tag{E.4}$$

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma^2}} = (\sigma^2 - \sigma_\mathrm{min}^2) \operatorname{trace}(\sigma^{-2} - \mathbf{D_A}) \; , \tag{E.5}$$

also in $\mathcal{O}(m^2 n)$ time.

Derivatives wrt the remaining hyperparameteres can be computed using

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta} = -\frac{\sigma_0^2}{m\sigma_\mathrm{p}^2} \operatorname{trace}\left(\mathbf{F}_\mathbf{\Phi}^\top \frac{\partial \mathbf{\Phi_f}}{\partial \theta}\right) \; . \tag{E.6}$$

Note that these latter derivatives can be computed in $\mathcal{O}(mn)$ time each, so that a total of $\mathcal{O}(m)$ derivatives can be computed without increasing the complexity of the procedure. Usually, derivative matrix $\frac{\partial \mathbf{\Phi_f}}{\partial \theta}$ has a single non-zero row, so computation time takes only $\mathcal{O}(n)$ time per derivative. Concrete expressions for $\frac{\partial \mathbf{\Phi_f}}{\partial \theta}$ corresponding to the MN models in this thesis follow.

E.2.1 SSGP design matrix derivatives

First we define the common part of the derivatives

$$\mathbf{C}^\top = \begin{bmatrix} -\sin(2\pi \mathbf{s}_1^\top \mathbf{x}_1) & \cos(2\pi \mathbf{s}_1^\top \mathbf{x}_1) & \dots & -\sin(2\pi \mathbf{s}_h^\top \mathbf{x}_1) & \cos(2\pi \mathbf{s}_h^\top \mathbf{x}_1) \\ -\sin(2\pi \mathbf{s}_1^\top \mathbf{x}_2) & \cos(2\pi \mathbf{s}_1^\top \mathbf{x}_2) & \dots & -\sin(2\pi \mathbf{s}_h^\top \mathbf{x}_2) & \cos(2\pi \mathbf{s}_h^\top \mathbf{x}_2) \\ \vdots & & \ddots & & \vdots \\ -\sin(2\pi \mathbf{s}_1^\top \mathbf{x}_n) & \cos(2\pi \mathbf{s}_1^\top \mathbf{x}_n) & \dots & -\sin(2\pi \mathbf{s}_h^\top \mathbf{x}_n) & \cos(2\pi \mathbf{s}_h^\top \mathbf{x}_n) \end{bmatrix}_{n \times 2h} ,$$

and then compute the derivatives wrt the length-scales and (unscaled) spectral points as

$$\left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial \theta_{\ell_d}}\right]_{ij} = \begin{cases} -[\mathbf{C}]_{ij} 2\pi [\mathbf{x}_j]_d [\mathbf{s}_{\frac{i+1}{2}}]_d & i \text{ odd} \\ -[\mathbf{C}]_{ij} 2\pi [\mathbf{x}_j]_d [\mathbf{s}_{\frac{i}{2}}]_d & i \text{ even} \end{cases} \tag{E.7}$$

$$\left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial [\boldsymbol{\omega}_r]_d}\right]_{ij} = \begin{cases} [\mathbf{C}]_{ij} 2\pi [\mathbf{x}_j]_d / \ell_d & i/2 = r \text{ or } (i+1)/2 = r \\ 0 & \text{otherwise} \end{cases} \tag{E.8}$$

where, as usual, $j \in 1 \dots n$, $i \in 1 \dots m$, and $r \in 1 \dots h$, with $m = 2h$. For any given r, the latter derivative will only have two non-zero rows.

E.2.2 MCN design matrix derivatives (also BN-MCN, MCNmix)

Again, we define the common part of the derivatives as

$$a_{ij} = -\sin(\tilde{\mathbf{x}}_j^\top \mathbf{u}_i) \quad \text{(recall that } \mathbf{u}_i = [\varphi_i,\ (\mathbf{L}^{-1}\boldsymbol{\omega}_i)^\top]^\top\text{)}$$

and then compute the derivatives wrt the length-scales and (unscaled) input weights as

$$\left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial \theta_{\ell_d}}\right]_{ij} = -a_{ij} [\tilde{\mathbf{x}}_j]_d [\mathbf{u}_i]_d \ ; \qquad \left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial [\boldsymbol{\omega}_{i'}]_d}\right]_{ij} = a_{ij} [\tilde{\mathbf{x}}_j]_d / \ell_d \delta_{ii'} \ , \tag{E.9}$$

where Kronecker delta $\delta_{ii'}$ equals one when $i = i'$ and zero otherwise. Only the i'-th row of $\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial [\boldsymbol{\omega}_{i'}]_d}$ is different from zero.

Analogously, the derivatives wrt the phase hyperparameters are

$$\left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial \varphi_{i'}}\right]_{ij} = a_{ij} \delta_{ii'} \ . \tag{E.10}$$

E.2.3 MMLP design matrix derivatives (also BN-MMLP, MMLPmix)

All derivatives share a common term

$$a_{ij} = \frac{2}{\sqrt{\pi}} \exp(-(\tilde{\mathbf{x}}_j^\top \mathbf{u}_i)^2) \quad \text{(recall that for MMLPs, } \mathbf{u}_i = \tilde{\mathbf{L}}^{-1} \boldsymbol{\omega}_i\text{)}.$$

The derivatives of the design matrix wrt the length-scales and (unscaled) input weights are

$$\left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial \theta_{\ell_d}}\right]_{ij} = -a_{ij} [\tilde{\mathbf{x}}_j]_d [\mathbf{u}_i]_d \ ; \qquad \left[\frac{\partial \mathbf{\Phi}_\mathbf{f}}{\partial [\boldsymbol{\omega}_{i'}]_d}\right]_{ij} = a_{ij} [\tilde{\mathbf{x}}_j]_d / \ell_d \delta_{ii'} \ , \tag{E.11}$$

where Kronecker delta $\delta_{ii'}$ equals one when $i = i'$ and zero otherwise. Only the i'-th row of $\frac{\partial \mathbf{\Phi_f}}{\partial [\boldsymbol{\omega}_{i'}]_d}$ is different from zero.

E.3 Inter-Domain GP log-evidence derivatives

We consider $\mathbf{R_{uu}}$, $\mathbf{\Lambda}$, $\mathbf{\Phi}_\mathrm{s}$, $\mathbf{y}_\mathrm{s}$, $\mathbf{H}$, and $\boldsymbol{\beta}$ as defined by (D.4) and precompute the following matrices:

$$\mathbf{b} = (\mathbf{y}_\mathrm{s} - \mathbf{H}^\top \boldsymbol{\beta})/\sigma \tag{E.12}$$

$$\mathbf{A} = [\mathbf{H}^\top,\ \mathbf{b}]^\top \mathbf{\Lambda}^{-1/2}\ ; \quad \mathbf{D_A} = \mathrm{diag}(\mathbf{A}^\top \mathbf{A})\ ; \quad \mathbf{D_\Phi} = \mathrm{diag}(\mathbf{\Phi}_\mathrm{s}^\top \mathbf{\Phi}_\mathrm{s}) \tag{E.13}$$

$$\mathbf{F_{fu}^\top} = \mathbf{R_{uu}} \backslash ((\mathbf{\Phi}_\mathrm{s}\mathbf{A}^\top)\mathbf{A} - \mathbf{\Phi}_\mathrm{s}\mathbf{D_A})/\sigma^2 \tag{E.14}$$

$$\mathbf{F_{uu}} = -\mathbf{R_{uu}} \backslash (\mathbf{\Phi}_\mathrm{s}\mathbf{F_{fu}})\ . \tag{E.15}$$

The derivatives of the NLML wrt power hyperparameters (defined in Section E.1) are:

$$\begin{aligned} -\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma_0^2}} &= (\sigma_0^2 \,\mathrm{trace}(\mathbf{\Lambda}^{-1} - \mathbf{D_A}) + \mathrm{trace}(\mathbf{A}\mathbf{D_\Phi}\mathbf{A}^\top) \\ &\quad - \mathrm{trace}((\mathbf{\Phi}_\mathrm{s}\mathbf{A}^\top)(\mathbf{\Phi}_\mathrm{s}\mathbf{A}^\top)^\top))\sigma^{-2} \end{aligned} \tag{E.16}$$

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma^2}} = \mathrm{trace}(\mathbf{\Lambda}^{-1} - \mathbf{D_A})\ . \tag{E.17}$$

All the above operations take $\mathcal{O}(m^2 n)$ time. If we express the derivative of the transformed domain covariance matrix wrt the remaining hyperparameters as

$$\frac{\partial \mathbf{K_{uu}}}{\partial \theta} = \left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta}\right]_\mathrm{half} + \left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta}\right]_\mathrm{half}^\top , \tag{E.18}$$

(which is possible, since it is symmetric), the derivatives of the NLML wrt the remaining hyperparameters can be computed using

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta} = -\,\mathrm{trace}\left(\mathbf{F_{fu}^\top}\frac{\partial \mathbf{K_{fu}}}{\partial \theta} + \mathbf{F_{uu}}\left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta}\right]_\mathrm{half}\right)\ , \tag{E.19}$$

which takes $\mathcal{O}(mn)$ time. Therefore, $\mathcal{O}(m)$ derivatives can be computed without increasing the overall complexity of the procedure. For our models, many of these operations only take $\mathcal{O}(n)$ time because $\frac{\partial \mathbf{K_{fu}}}{\partial \theta}$ and $\left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta}\right]_\mathrm{half}$ have a single non-zero row per derivative. Expressions for the derivatives of the inter-domain and transformed domain prior covariance matrices are given in the following subsections.

E.3.1 TFIFGP prior covariance derivatives (also FIFGP)

Here we describe the derivatives of the inter-domain and transformed domain instances of TFIFGP's covariance. Instead of directly optimizing window length-scales $\{c_d\}_{d=1}^D$ (which must be positive), we we will prefer to optimize their logarithms $\theta_{c_d} = \log(c_d)$ (which can take any real value), so that we can use unconstrained optimization algorithms. Therefore, derivatives wrt $\{\theta_{c_d}\}_{d=1}^D$ are provided. The same is done for power hyperparameters and length-scales, see Section E.1.

We consider inter-domain covariance $\mathbf{K_{fu}}$ first. Defining the auxiliary values

$$a_{ji} = \exp\left[-\sum_{d=1}^{D} \frac{([\mathbf{x}_j]_d - [\boldsymbol{\mu}_i]_d)^2 + c_d^2[\boldsymbol{\omega}_i]_d^2}{2(c_d^2 + \ell_d^2)}\right] \tag{E.20}$$

$$b_{ji} = [\boldsymbol{\omega}_i]_0 + \sum_{d=1}^{D} \frac{c_d^2[\boldsymbol{\omega}_i]_d([\mathbf{x}_j]_d - [\boldsymbol{\mu}_i]_d)}{c_d^2 + \ell_d^2} \tag{E.21}$$

$$v = \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{c_d^2 + \ell_d^2}}\,, \tag{E.22}$$

it is possible to compute the derivatives wrt all hyperparameters as

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial \theta_{\ell_d}}\right]_{ji} = a_{ji} v \left(\left(\frac{c_d^2 v^2}{\ell_d^2} + \frac{c_d^2 \ell_d^2 [\boldsymbol{\omega}_i]_d^2}{(c_d^2 + \ell_d^2)^2}\right)\cos(b_{ji}) + \frac{2c_d^2 \ell_d^2 [\boldsymbol{\omega}_i]_d([\mathbf{x}_j]_d - [\boldsymbol{\mu}_i]_d)}{(c_d^2 + \ell_d^2)^2}\sin(b_{ji})\right) \tag{E.23}$$

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial \theta_{c_d}}\right]_{ji} = -\left[\frac{\partial \mathbf{K_{fu}}}{\partial \theta_{\ell_d}}\right]_{ji} \tag{E.24}$$

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial [\boldsymbol{\mu}_{i'}]_d}\right]_{ji} = a_{ji} v \frac{([\mathbf{x}_j]_d - [\boldsymbol{\mu}_i]_d)\cos(b_{ji}) + c_d^2[\boldsymbol{\omega}_i]_d \sin(b_{ji})}{c_d^2 + \ell_d^2}\delta_{ii'} \tag{E.25}$$

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial [\boldsymbol{\omega}_{i'}]_d}\right]_{ji} = a_{ji} v c_d^2 \frac{([\mathbf{x}_j]_d - [\boldsymbol{\mu}_i]_d)\sin(b_{ji}) + [\boldsymbol{\omega}_i]_d \cos(b_{ji})}{c_d^2 + \ell_d^2}\delta_{ii'} \tag{E.26}$$

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial [\boldsymbol{\omega}_{i'}]_0}\right]_{ji} = -a_{ji} v \sin(b_{ji})\delta_{ii'} \tag{E.27}$$

in $\mathcal{O}(m^2 n)$ time. Note for that the last three derivatives, only the i'-th column of the matrix needs to be computed.

Similarly, for intra-domain covariance $\mathbf{K_{uu}}$, we define

$$a_{ji} = \exp\left[-\sum_{d=1}^{D} \frac{(c_d^4 + c_d^2)([\boldsymbol{\omega}_i]_d^2 + [\boldsymbol{\omega}_j]_d^2) + ([\boldsymbol{\mu}_i]_d - [\boldsymbol{\mu}_j]_d)^2}{2(2c_d^2 + \ell_d^2)}\right] \tag{E.28}$$

$$b_{ji} = \exp\left[\sum_{d=1}^{D} \frac{c_d^4 [\boldsymbol{\omega}_i]_d [\boldsymbol{\omega}_j]_d}{2c_d^2 + \ell_d^2}\right] \tag{E.29}$$

$$v = \prod_{d=1}^{D} \sqrt{\frac{\ell_d^2}{2c_d^2 + \ell_d^2}} \tag{E.30}$$

$$g_{ji} = b_{ji} \cos([\boldsymbol{\omega}_i]_0 - [\boldsymbol{\omega}_j]_0) + b_{ji}^{-1} \cos([\boldsymbol{\omega}_i]_0 + [\boldsymbol{\omega}_j]_0) \tag{E.31}$$

$$g'_{ji} = b_{ji} \cos([\boldsymbol{\omega}_i]_0 - [\boldsymbol{\omega}_j]_0) - b_{ji}^{-1} \cos([\boldsymbol{\omega}_i]_0 + [\boldsymbol{\omega}_j]_0)\ , \tag{E.32}$$

and then compute one "half" of the derivatives

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta_{\ell_d}}\right]_{\text{half},\, ji} = \frac{1}{2} a_{ji} v \left(g_{ji} \left(\frac{\ell_d^2((c_d^4 + c_d^2)([\boldsymbol{\omega}_i]_d^2 + [\boldsymbol{\omega}_j]_d^2) + ([\boldsymbol{\mu}_i]_d - [\boldsymbol{\mu}_j]_d)^2)}{(2c_d^2 + \ell_d^2)^2} + \frac{c_d^2 v^2}{\ell_d^2} \right) + g'_{ji} \frac{2\ell_d^2 c_d^4 [\boldsymbol{\omega}_i]_d [\boldsymbol{\omega}_j]_d}{(2c_d^2 + \ell_d^2)^2} \right) \tag{E.33}$$

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta_{c_d}}\right]_{\text{half},\, ji} = \frac{1}{2} a_{ji} v \left(g_{ji} \left(\frac{c_d^2(2(c_d^4 + c_d^2) - (2c_d^2 + \ell_d^2)(2c_d^2 + 1))([\boldsymbol{\omega}_i]_d^2 + [\boldsymbol{\omega}_j]_d^2)}{(2c_d^2 + \ell_d^2)^2} - \frac{c_d^2 v^2}{\ell_d^2} + \frac{2c_d^2([\boldsymbol{\mu}_i]_d - [\boldsymbol{\mu}_j]_d)^2}{(2c_d^2 + \ell_d^2)^2} \right) + g'_{ji} \frac{4(c_d^2 + \ell_d^2) c_d^4 [\boldsymbol{\omega}_i]_d [\boldsymbol{\omega}_j]_d}{(2c_d^2 + \ell_d^2)^2} \right) \tag{E.34}$$

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial [\boldsymbol{\mu}_{i'}]_d}\right]_{\text{half},\, ji} = a_{ji} g_{ji} v \frac{-([\boldsymbol{\mu}_i]_d - [\boldsymbol{\mu}_j]_d)}{2c_d^2 + \ell_d^2} \delta_{ii'} = \frac{-([\boldsymbol{\mu}_i]_d - [\boldsymbol{\mu}_j]_d)}{2c_d^2 + \ell_d^2} [\mathbf{K_{uu}}]_{ji} \delta_{ii'} \tag{E.35}$$

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial [\boldsymbol{\omega}_{i'}]_d}\right]_{\text{half},\, ji} = a_{ji} v \left(-g_{ji} \frac{[\boldsymbol{\omega}_i]_d (c_d^4 + c_d^2)}{2c_d^2 + \ell_d^2} + g'_{ji} \frac{c_d^4}{2c_d^2 + \ell_d^2} \right) \delta_{ii'} \tag{E.36}$$

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial [\boldsymbol{\omega}_{i'}]_0}\right]_{\text{half},\, ji} = -a_{ji} v (b_{ji} \sin([\boldsymbol{\omega}_i]_0 - [\boldsymbol{\omega}_j]_0) + b_{ji}^{-1} \sin([\boldsymbol{\omega}_i]_0 + [\boldsymbol{\omega}_j]_0)) \delta_{ii'}\ . \tag{E.37}$$

Again, for the last three derivatives, all entries are zero except those in column i'. Full derivatives of $\mathbf{K_{uu}}$, though not needed, can be obtained combining these equations with (E.18). Derivatives for FIFGP are obtained by letting $\{\boldsymbol{\mu}_i\}_{i=1}^m = \mathbf{0}$ in (E.20)-(E.37).

E.3.2 SPGP ARD MLP prior covariance derivatives

We limit ourselves here to the case of the ARD MLP covariance function. Derivatives for the more common ARD SE covariance function are detailed in Snelson (2007, App. C).

For covariance matrix $\mathbf{K_{fu}}$ we first compute

$$\hat{\mathbf{x}}_j = \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{x}}_j \; ; \qquad \hat{\mathbf{u}}_i = \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{u}}_i \tag{E.38}$$

$$a_{ji} = 2\hat{\mathbf{x}}_j^\top \hat{\mathbf{u}}_i \; ; \qquad b_j = (1 + 2\hat{\mathbf{x}}_j^\top \hat{\mathbf{x}}_j)^{-\frac{1}{2}} \; ; \qquad c_i = (1 + 2\hat{\mathbf{u}}_i^\top \hat{\mathbf{u}}_i)^{-\frac{1}{2}} \tag{E.39}$$

$$g_{ji} = \frac{4\sigma_0^2 b_j c_i}{\pi\sqrt{1-(a_{ji}b_j c_i)^2}} \; , \tag{E.40}$$

so that we can obtain the derivatives wrt length-scales and pseudo-inputs as

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial \theta_{\ell_d}}\right]_{ji} = g_{ji}(a_{ji}(b_j^2[\hat{\mathbf{x}}_j]_d^2 + c_i^2[\hat{\mathbf{u}}_i]_d^2) - 2[\hat{\mathbf{x}}_j]_d[\hat{\mathbf{u}}_i]_d) \tag{E.41}$$

$$\left[\frac{\partial \mathbf{K_{fu}}}{\partial [\mathbf{u}_{i'}]_d}\right]_{ji} = \frac{1}{\ell_d} g_{ji}([\hat{\mathbf{x}}_j]_d - a_{ji}c_i^2[\hat{\mathbf{u}}_i]_d)\delta_{ii'} \; . \tag{E.42}$$

For covariance matrix $\mathbf{K_{uu}}$ we redefine

$$a_{ji} = 2\hat{\mathbf{u}}_j^\top \hat{\mathbf{u}}_i \tag{E.43}$$

$$g_{ji} = \frac{4\sigma_0^2 c_j c_i}{\pi\sqrt{1-(a_{ji}c_j c_i)^2}} \; , \tag{E.44}$$

and consider the same c_i as before. Then we obtain one "half"of the derivatives wrt the length-scales and pseudo-inputs:

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial \theta_{\ell_d}}\right]_{\text{half},\, ji} = \frac{1}{2} g_{ji}(a_{ji}(c_j^2[\hat{\mathbf{u}}_j]_d^2 + c_i^2[\hat{\mathbf{u}}_i]_d^2) - 2[\hat{\mathbf{u}}_j]_d[\hat{\mathbf{u}}_i]_d) \tag{E.45}$$

$$\left[\frac{\partial \mathbf{K_{uu}}}{\partial [\mathbf{u}_{i'}]_d}\right]_{\text{half},\, ji} = \frac{1}{\ell_d} g_{ji}([\hat{\mathbf{u}}_j]_d - a_{ji}c_i^2[\hat{\mathbf{u}}_i]_d)\delta_{ii'} \; . \tag{E.46}$$

Last derivative matrix has a single non-zero column. Full derivatives can be obtained using (E.18).

E.4 Non-Gaussian log-evidence derivatives

Derivatives described in Sections E.2 and E.3 apply to standard MNs and IDGPs using a Gaussian likelihood. In the following we detail the derivatives for these models when used with non-Gaussian likelihoods.

E.4.1 Log-evidence derivatives for IDGP with non-Gaussian likelihood

In regression models, latent function $f(\mathbf{x})$ is usually regarded as noiseless, and noise is introduced through a Gaussian likelihood function. When dealing with non-Gaussian likelihoods, it may be convenient to also include Gaussian noise in the latent function. This is the case in classification, see Subsection 5.4.1. Therefore, in the derivations below we consider Gaussian noise with power σ^2. Inducing variables $\{\mathbf{u}_i\}_{i=1}^m$ remain noiseless.

Define $\mathbf{\Lambda_f}$ as an $n \times n$ diagonal matrix, with j-th element of the diagonal being $\sigma^2 + k(\mathbf{x}_j, \mathbf{x}_j) + \mathbf{k_u}(\mathbf{x}_j)^\top \mathbf{K_{uu}^{-1}} \mathbf{k_u}(\mathbf{x}_j)$. Here a noiseless $k(\cdot,\cdot)$ is used and noise is added explicitly. Consider $\mathbf{R_{uu}}$, and $\mathbf{\Phi_f}$ defined as per (D.4). Then compute the following matrices

$$\tilde{\mathbf{\Lambda}}^{-1} = \tilde{\mathbf{T}}(\mathbf{I}_n + \mathbf{\Lambda_f}\tilde{\mathbf{T}})^{-1} \; ; \quad \mathbf{\Phi}_\mathrm{s} = \mathbf{\Phi_f}\tilde{\mathbf{\Lambda}}^{-1/2} \tag{E.47}$$

$$\mathbf{R} = \mathrm{chol}(\mathbf{I}_m + \mathbf{\Phi}_\mathrm{s}\mathbf{\Phi}_\mathrm{s}^\top) \; ; \quad \mathbf{H} = \mathbf{R}^\top \backslash (\mathbf{\Phi}_\mathrm{s}\tilde{\mathbf{\Lambda}}^{-1/2}) \tag{E.48}$$

$$\mathbf{c} = \mathbf{\Phi_f}^\top(\mathbf{\Phi_f}\tilde{\boldsymbol{\nu}}) + \mathbf{\Lambda_f}\tilde{\boldsymbol{\nu}} \; ; \quad \mathbf{b} = \tilde{\boldsymbol{\nu}} - (\tilde{\mathbf{\Lambda}}^{-1}\mathbf{c} - (\mathbf{H}^\top(\mathbf{H}\mathbf{c}))) \tag{E.49}$$

$$\mathbf{A} = [\mathbf{H}^\top, \, \mathbf{b}]^\top \; ; \quad \mathbf{D_A} = \mathrm{diag}(\mathbf{A}^\top\mathbf{A}) \; ; \quad \mathbf{D_\Phi} = \mathrm{diag}(\mathbf{\Phi_f}^\top\mathbf{\Phi_f}) \tag{E.50}$$

$$\mathbf{F_{fu}}^\top = \mathbf{R_{uu}} \backslash ((\mathbf{\Phi_f}\mathbf{A}^\top)\mathbf{A} - \mathbf{\Phi_f}\mathbf{D_A}) \tag{E.51}$$

$$\mathbf{F_{uu}} = -\mathbf{R_{uu}} \backslash (\mathbf{\Phi_f}\mathbf{F_{fu}}) \; , \tag{E.52}$$

where $\tilde{\mathbf{T}}$ and $\tilde{\boldsymbol{\nu}}$ are the site parameters after EP convergence.

Derivatives of the approximate NLML wrt the power hyperparameters can be ob-

tained with

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma_0^2}} = \sigma_0^2 \operatorname{trace}(\tilde{\mathbf{\Lambda}}^{-1} - \mathbf{D}_{\mathbf{A}}) + \operatorname{trace}(\mathbf{A}\mathbf{D}_{\mathbf{\Phi}}\mathbf{A}^\top) \tag{E.53}$$

$$- \operatorname{trace}((\mathbf{\Phi}_\mathbf{f}\mathbf{A}^\top)(\mathbf{\Phi}_\mathbf{f}\mathbf{A}^\top)^\top) \tag{E.54}$$

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma^2}} = \sigma^2 \operatorname{trace}(\tilde{\mathbf{\Lambda}}^{-1} - \mathbf{D}_{\mathbf{A}}) \tag{E.55}$$

All the above operations take $\mathcal{O}(m^2 n)$ time. Expressing the derivative of the transformed domain covariance matrix according to (E.18), the derivatives of the approximate NLML wrt the remaining hyperparameters can be computed using

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta} = -\operatorname{trace}\left(\mathbf{F}_{\mathbf{fu}}^\top \frac{\partial \mathbf{K}_{\mathbf{fu}}}{\partial \theta} + \mathbf{F}_{\mathbf{uu}}\left[\frac{\partial \mathbf{K}_{\mathbf{uu}}}{\partial \theta}\right]_{\text{half}}\right) , \tag{E.56}$$

which takes $\mathcal{O}(mn)$ time. As noted in Section E.3, this allows to compute $\mathcal{O}(m)$ derivatives without increasing the overall complexity of the procedure.

E.4.1.1 FIFGPC log-evidence derivatives

The approximate NLML derivatives for the FIFGPC model from Section 4.3.4 can be obtained following the above procedure and plugging the FIFGP covariance derivatives from Section E.3.1 in (E.56).

E.4.2 Log-evidence derivatives for MNs with non-Gaussian likelihood

As above, we will consider the general case in which a noisy latent function with noise power σ^2 is used. Compute

$$\tilde{\mathbf{\Lambda}}^{-1} = \tilde{\mathbf{T}}(\mathbf{I}_n + \sigma^2\tilde{\mathbf{T}})^{-1} ; \quad \mathbf{\Phi}_\text{s} = \mathbf{\Phi}_\mathbf{f}\tilde{\mathbf{\Lambda}}^{-1/2} \tag{E.57}$$

$$\mathbf{R} = \operatorname{chol}(\mathbf{I}_m + \mathbf{\Phi}_\text{s}\mathbf{\Phi}_\text{s}^\top) ; \quad \mathbf{H} = \mathbf{R}^\top \backslash (\mathbf{\Phi}_\text{s}\tilde{\mathbf{\Lambda}}^{-1/2}) \tag{E.58}$$

$$\mathbf{c} = \mathbf{\Phi}_\mathbf{f}^\top(\mathbf{\Phi}_\mathbf{f}\tilde{\boldsymbol{\nu}}) + \sigma^2\tilde{\boldsymbol{\nu}} ; \quad \mathbf{b} = \tilde{\boldsymbol{\nu}} - (\tilde{\mathbf{\Lambda}}^{-1}\mathbf{c} - (\mathbf{H}^\top(\mathbf{H}\mathbf{c}))) \tag{E.59}$$

$$\mathbf{A} = [\mathbf{H}^\top, \mathbf{b}]^\top ; \quad \mathbf{D}_\mathbf{A} = \operatorname{diag}(\mathbf{A}^\top\mathbf{A}) ; \quad \mathbf{D}_\mathbf{\Phi} = \operatorname{diag}(\mathbf{\Phi}_\mathbf{f}^\top\mathbf{\Phi}_\mathbf{f}) \tag{E.60}$$

$$\mathbf{F}_\mathbf{\Phi}^\top = (\mathbf{\Phi}_\mathbf{f}\mathbf{A}^\top)\mathbf{A} - \mathbf{\Phi}_\mathbf{f}\tilde{\mathbf{\Lambda}}^{-1} . \tag{E.61}$$

where $\tilde{\mathbf{T}}$ and $\tilde{\boldsymbol{\nu}}$ are the site parameters after EP convergence. After these $\mathcal{O}(m^2n)$ precomputations, derivatives of the approximate NLML wrt power hyperparameters (as defined in Section E.1) can be obtained from

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma_0^2}} = \frac{\sigma_0^2}{m\sigma_\text{p}^2}(\text{trace}(\mathbf{D}_{\boldsymbol{\Phi}}\tilde{\boldsymbol{\Lambda}}^{-1}) - \text{trace}((\boldsymbol{\Phi}_\mathbf{f}\mathbf{A}^\top)(\boldsymbol{\Phi}_\mathbf{f}\mathbf{A}^\top)^\top)) \tag{E.62}$$

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma^2}} = (\sigma^2 - \sigma_\text{min}^2)\,\text{trace}(\tilde{\boldsymbol{\Lambda}}^{-1} - \mathbf{D}_\mathbf{A})\,, \tag{E.63}$$

also in $\mathcal{O}(m^2n)$ time.

Derivatives wrt the remaining hyperparameteres can be computed using

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta} = -\frac{\sigma_0^2}{m\sigma_\text{p}^2}\,\text{trace}\left(\mathbf{F}_{\boldsymbol{\Phi}}^\top \frac{\partial \boldsymbol{\Phi}_\mathbf{f}}{\partial \theta}\right)\,, \tag{E.64}$$

which takes $\mathcal{O}(mn)$ time. Therefore, when computing $\mathcal{O}(m)$ derivatives, the whole procedure will take $\mathcal{O}(m^2n)$ time.

E.4.2.1 Robust BN-MCN log-evidence derivatives

The NLML derivatives of the robust BN-MCN model from Section 5.3.2 can be computed using the above equations. For (E.64), the covariance matrix derivatives from Section E.2.2 must be used. Gaussian noise power σ_0^2 can be set to 0 (as we did in our experiments) or learned, thus modeling a Gaussian noise term in addition to the Laplace noise term.

Robust regression with Laplace noise has an additional hyperparameter, σ_L, which is not part of the covariance, but of the likelihood function. Instead of directly learning σ_L, we will learn its logarithm $\theta_{\sigma_\text{L}^2} = \frac{1}{2}\log(\sigma_\text{L}^2)$, which is not constrained to be positive, and therefore amenable to conjugate gradient optimization. The derivative of the approximate NLML wrt this hyperparameter is

$$-\frac{\partial \log q(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma_\text{L}^2}} = -(\sigma_\text{L}^2 - \sigma_\text{Lmin}^2)\sum_{j=1}^{n}\frac{g_j}{4\hat{m}_{0j}\sigma_\text{L}^2}\,, \tag{E.65}$$

where $\hat{m}_{0j}$ is defined by (5.38) and g_j is

$$g_j = \sqrt{\frac{8\hat{\sigma}_{\backslash j}^2}{\pi}}\exp\left(-\frac{\hat{\mu}_{\backslash j}^2}{2\hat{\sigma}_{\backslash j}^2}\right) - \exp\left(\frac{\hat{\sigma}_{\backslash j}^2}{2} - \hat{\mu}_{\backslash j}\right)((1+\hat{\sigma}_{\backslash j}^2 - \hat{\mu}_{\backslash j})a_j + (1+\hat{\sigma}_{\backslash j}^2 + \hat{\mu}_{\backslash j})b_j)\,,$$

in turn parametrized by $\hat{\mu}_{\backslash j}$, $\hat{\sigma}_{\backslash j}^2$, a_j and b_j, which are defined by (5.39).

E.5 Full GP log-evidence derivatives

For standard regression GP models with covariance matrix $\mathbf{K_{ff}}+\sigma^2\mathbf{I}_n$ and observations $\mathbf{y}$, the derivative wrt the noise power hyperparameter can be computed as

$$\mathbf{R} = \mathrm{chol}(\mathbf{K_{ff}} + \sigma^2\mathbf{I}_n) \qquad \mathbf{H} = \mathbf{R}^\top\backslash\mathbf{I}_n \qquad \boldsymbol{\alpha} = \mathbf{R}\backslash(\mathbf{R}^\top\backslash\mathbf{y}) \tag{E.66}$$

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta_{\sigma^2}} = \frac{\sigma^2}{2}(||\boldsymbol{\alpha}||^2 - ||\mathbf{H}||^2_{\text{Frob}}) \tag{E.67}$$

and wrt the remaining hyperparameters as

$$\mathbf{R}_{\text{der}} = \mathrm{chol}\left(\frac{\partial \mathbf{K_{ff}}}{\partial \theta}\right) \tag{E.68}$$

$$-\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \theta} = \frac{1}{2}(||\mathbf{R}_{\text{der}}\boldsymbol{\alpha}||^2 - ||\mathbf{R}_{\text{der}}\mathbf{H}||^2_{\text{Frob}}) \ , \tag{E.69}$$

where $||\cdot||_{\text{Frob}}$ stands for the Frobenius norm.

The cost of this procedure is dominated by the Cholesky factorizations that take $\mathcal{O}(n^3)$ time (per hyperparameter).

Appendix F

Code

Code implementing the described algorithms can be found online:

http://www.tsc.uc3m.es/~miguel/

Bibliography

Alvarez, M. and Lawrence, N. D. (2009). Sparse convolved Gaussian processes for multi-output regression. In *Advances in Neural Information Processing Systems 21*, pages 57–64. 91

Baker, C. T. H. (1977). *The Numerical treatment of integral equations*. Clarendon Press. 13

Bonilla, E., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems 20*, pages 153–160. MIT Press. xix, 154

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 26:123–140. xv, 74

Bretthorst, G. L. (2000). Nonuniform sampling: Bandwidth and aliasing. In *Maximum Entropy and Bayesian Methods*, pages 1–28. Kluwer. 34

Bryson, A. E. and Ho, Y. C. (1969). *Applied optimal control: optimization, estimation, and control*. Blaisdell Publishing Company. 107

Carlson, A. B. (1986). *Communication Systems*. McGraw-Hill, 3rd edition. 24

Chen, T. and Ren, J. (2009). Bagging for Gaussian process regression. *Neurocomputing*, 72:1605–1610. 74

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297. 144

Csató, L. and Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, 14(3):641–669. 12, 16, 19, 94

Each reference is followed by the page numbers where the corresponding citation appears.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314. 106

DeGroot, M. H. and Schervish, M. J. (2002). *Probability and Statistics*. Addison-Wesley. 123

Gibbs, M. N. (1997). Bayesian Gaussian processes for regression and classification. *PhD thesis*. 20

Golub, G. H. and Loan, C. F. V. (1989). *Matrix Computations*. John Hopkins University Press. 157

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspective*. Springer. 157

Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks*, 6:1069–1072. 106

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366. 106

Kuss, M. (2006). *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt. 119, 120, 132, 133

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704. 119

Kuss, M. and Rasmussen, C. E. (2006). Assessing approximations for Gaussian process classification. In *Advances in Neural Information Processing Systems 18*, pages 699–706. MIT Press. 119

Lawrence, N. D., Seeger, M., and Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press. 13, 144

Lazaro-Gredilla, M., Candela, J. Q., and Figueiras-Vidal, A. (2007). Sparse spectral sampling Gaussian processes. Technical report, Microsoft Research. 23

Lázaro-Gredilla, M. and Figueiras-Vidal, A. (2010). Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems 22*, pages 1087–1095. MIT Press. 89

Lütkepohl, H. (1996). *Handbook of Matrices*. John Wiley & Sons. 157

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. 28

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press. 159

Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in Applied Probability*, 5:439–468. 2

Minka, T. P. (2001). *Expectation Propagation for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology. 119

Naish-Guzman, A. and Holden, S. (2008). The generalized FITC approximation. In *Advances in Neural Information Processing Systems 20*, pages 1057–1064. MIT Press. 1, 127, 130, 144, 147

Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical report, University of Toronto. 72

Neal, R. M. (1993). Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems 5*, pages 475–482. Morgan Kaufmann. 72

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag. 72, 108

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society*, 40:1–42. 2

Potgietera, G. and Engelbrecht, A. P. (2002). Pairwise classification as an ensemble technique. In *Proceedings of the 13th European Conference on Machine Learning*, pages 97–110. Springer-Verlag. 39

Potgietera, G. and Engelbrecht, A. P. (2007). Evolving model trees for mining data sets with continuous-valued classes. *Expert Systems with Applications*, 35:1513–1532. 39

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2002). *Numerical Recipes in C++*. Cambridge University Press. 157

Qi, Y., Minka, T., and Picard, R. (2002). Bayesian spectrum estimation of unevenly sampled nonstationary data. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, volume 2, pages 1473–1476. 34

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959. 13, 16, 17, 18, 20, 126

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press, Cambridge, MA. 23

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and other Methods for Non-linear Regression*. PhD thesis, University of Toronto. 1

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 63–71. Springer. 10

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press. 2, 16, 120, 124, 125, 139, 142

Rätsch, G., Onoda, T., and Müler, K. R. (2001). Soft margins for Adaboost. *Machine Learning*, 42:287–320. 144

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the 9th International Workshop on AI Stats*. 12, 13, 16, 17, 42, 68, 94

Shen, Y., Ng, A., and Seeger, M. (2006). Fast Gaussian process regression using kd-trees. In *Advances in Neural Information Processing Systems 18*, pages 1227–1234. MIT Press. 20

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society*, 47:1–52. 12, 15

Smola, A. J. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press. 12, 16, 94

Snelson, E. (2007). *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, University of Cambridge. 92, 179

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1259–1266. MIT Press. 12, 17, 19, 42, 43, 68, 89, 92, 94, 99

Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer-Verlag. 24

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the 12th International Workshop on AI Stats*. xix, 103, 155

Torgo, L. and da Costa, J. P. (2000). Clustered partial linear regression. In *Proceedings of the 11th European Conference on Machine Learning*, pages 426–436. Springer. 39

Tresp, V. (2000). A Bayesian committee machine. *Neural Computation*, 12:2719–2741. 12, 19, 94

Walder, C., Kim, K. I., and Schölkopf, B. (2008). Sparse multiscale Gaussian process regression. In *25th International Conference on Machine Learning*. ACM Press, New York. xii, xvi, 21, 89, 95, 97, 103, 151

Weiss, S. M. and Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403. 41

Wiener, N. (1949). *Bayesian Learning for Neural Networks*. MIT Press. 2

Williams, C. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351. 119

Williams, C. K. I. (1997). Computing with infinite networks. In *Advances in Neural Information Processing Systems 9*, pages 1069–1072. MIT Press. 55, 108

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*. MIT Press. 2

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press. 12, 13, 94

Yang, C., Duraiswami, R., and Davis, L. (2005). Efficient kernel machines using the improved fast Gauss transform. In *Advances in Neural Information Processing Systems 17*, pages 1561–1568. MIT Press. 20